FUTURE COMPUTING 2011

The Third International Conference on Future Computational Technologies and Applications

ISBN: 978-1-61208-154-0

September 25-30, 2011
Rome, Italy

FUTURE COMPUTING 2011 Editors

Kendall E. Nygard, North Dakota State University - Fargo, USA

Pascal Lorenz, University of Haute Alsace, France

FUTURE COMPUTING 2011

Foreword

The Third International Conference on Future Computational Technologies and Applications [FUTURE COMPUTING 2011], held between September 25 and 30, 2011 in Rome, Italy, targeted advanced computational paradigms and their applications. The focus was to cover (i) the advanced research on computational techniques that apply the newest human-like decisions, and (ii) applications on various domains. The new development led to special computational facets on mechanism-oriented computing, large-scale computing and technology-oriented computing. They are largely expected to play an important role in cloud systems, on-demand services, autonomic systems, and pervasive applications and services.

We take here the opportunity to warmly thank all the members of the FUTURE COMPUTING 2011 Technical Program Committee, as well as the numerous reviewers. The creation of such a broad and high quality conference program would not have been possible without their involvement. We also kindly thank all the authors who dedicated much of their time and efforts to contribute to FUTURE COMPUTING 2011. We truly believe that, thanks to all these efforts, the final conference program consisted of top quality contributions.

Also, this event could not have been a reality without the support of many individuals, organizations, and sponsors. We are grateful to the members of the FUTURE COMPUTING 2011 organizing committee for their help in handling the logistics and for their work to make this professional meeting a success.

We hope that FUTURE COMPUTING 2011 was a successful international forum for the exchange of ideas and results between academia and industry and for the promotion of progress in the field of future computational technologies and applications.

We are convinced that the participants found the event useful and communications very open. We also hope the attendees enjoyed the charm of Rome, Italy.

FUTURE COMPUTING 2011 Chairs:

Miriam A. M. Capretz, The University of Western Ontario - London, Canada
Marek J. Druzdzel, University of Pittsburgh, USA
Wolfgang Gentzsch, Expert HPC, Germany
Francesc Guim, Intel Corporation, Spain
Kendall E. Nygard, North Dakota State University - Fargo, USA
Hiroyuki Sato, The University of Tokyo, Japan
Cristina Seceleanu, Mälardalen University, Sweden
Vladimir Stantchev, Berlin Institute of Technology, Germany

FUTURE COMPUTING 2011

Committee

FUTURE COMPUTING Advisory Chairs

Cristina Seceleanu, Mälardalen University, Sweden
Hiroyuki Sato, The University of Tokyo, Japan
Miriam A. M. Capretz, The University of Western Ontario - London, Canada
Kendall E. Nygard, North Dakota State University - Fargo, USA
Vladimir Stantchev, Berlin Institute of Technology, Germany
Marek J. Druzdzel, University of Pittsburgh, USA

FUTURE COMPUTING 2011 Industry/Research

Francesc Guim, Intel Corporation, Spain
Wolfgang Gentzsch, Expert HPC, Germany

FUTURE COMPUTING 2011 Technical Program Committee

Radu Calinescu, Aston University, UK
Miriam A. M. Capretz, The University of Western Ontario - London, Canada
Massimiliano Caramia, University of Rome "Tor Vergata" - Rome, Italy
Key-Sun Choi, KAIST, Korea
Rodrigo de Barros Paes, Universidade Federal de Alagoas, Brazil
Leandro Dias da Silva, Federal University of Alagoas - Maceió, Brazil
Marek J. Druzdzel, University of Pittsburgh, USA / Bialystok Technical University, Poland
Francesco Fontanella, Università degli Studi di Cassino, Italy
Ivan Ganchev, University of Limerick, Ireland
Wolfgang Gentzsch, EU DEISA Project / Open Grid Forum, Germany
Victor Govindaswamy, Texas A&M University - Texarkana, USA
Michael Grottke, University of Erlangen-Nuremberg, Germany
Frances Guim, Intel Corporation, Spain
Muhammad Iftikhar, Universiti Malaysia Sabah (UMS), Malaysia
Dalia Kriksciuniene, Vilnius University, Lithuania
Carlos Léon de Mora, University of Seville, Spain
Lu Liu, University of Derby, UK
Panos M. Pardalos, University of Florida - Gainesville, USA
Constandinos Mavromoustakis, University of Nicosia, Cyprus
Susana Munoz Hernández, Universidad Politécnica de Madrid, Spain
Kendall E. Nygard, North Dakota State University - Fargo, USA
Prakash Ranganathan, University of North Dakota - Grand Forks, USA
Ivan Rodero, Rutgers the State University of New Jersey / NSF Center for Autonomic Computing, USA
Aitor Rodriguez, Autonoma University of Barcelona, Spain
Hiroyuki Sato, The University of Tokyo, Japan
Cristina Seceleanu, Mdh, Sweden
Vladimir Stantchev, Berlin Institute of Technology, Germany

Copyright Information

For your reference, this is the text governing the copyright release for material published by IARIA.

The copyright release is a transfer of publication rights, which allows IARIA and its partners to drive the dissemination of the published material. This allows IARIA to give articles increased visibility via distribution, inclusion in libraries, and arrangements for submission to indexes.

I, the undersigned, declare that the article is original, and that I represent the authors of this article in the copyright release matters. If this work has been done as work-for-hire, I have obtained all necessary clearances to execute a copyright release. I hereby irrevocably transfer exclusive copyright for this material to IARIA. I give IARIA permission or reproduce the work in any media format such as, but not limited to, print, digital, or electronic. I give IARIA permission to distribute the materials without restriction to any institutions or individuals. I give IARIA permission to submit the work for inclusion in article repositories as IARIA sees fit.

I, the undersigned, declare that to the best of my knowledge, the article is does not contain libelous or otherwise unlawful contents or invading the right of privacy or infringing on a proprietary right.

Following the copyright release, any circulated version of the article must bear the copyright notice and any header and footer information that IARIA applies to the published article.

IARIA grants royalty-free permission to the authors to disseminate the work, under the above provisions, for any academic, commercial, or industrial use. IARIA grants royalty-free permission to any individuals or institutions to make the article available electronically, online, or in print.

IARIA acknowledges that rights to any algorithm, process, procedure, apparatus, or articles of manufacture remain with the authors and their employers.

I, the undersigned, understand that IARIA will not be liable, in contract, tort (including, without limitation, negligence), pre-contract or other representations (other than fraudulent misrepresentations) or otherwise in connection with the publication of my work.

Exception to the above is made for work-for-hire performed while employed by the government. In that case, copyright to the material remains with the said government. The rightful owners (authors and government entity) grant unlimited and unrestricted permission to IARIA, IARIA's contractors, and IARIA's partners to further distribute the work.

Table of Contents

Efficient Swarm Algorithms to Constrained Resource Allocation

Peng-Yeng Yin
Department of Information Management
National Chi Nan University
Nantou, Taiwan
pyyin@ncnu.edu.tw

Jing-Yu Wang
Department of Information Management
National Chi Nan University
Nantou, Taiwan
wjykino@hotmail.com

Abstract—**Resource management is the effective deployment for an organization's resources. It deals with classification, allocation, stocking, processing, storage, and valuation for the shared resources when they are needed. Among many managerial tasks, the constrained resource allocation problem has challenging many practitioners involved in manufacturing operations. This problem seeks to find an optimal allocation of a limited amount of resource to a number of manufacturing activities for optimizing organization's objective subject to the resource constraints. Most existing methods use mathematical programming techniques, but they are defaced in deriving exact solutions for large-scale problems with reasonable time. A viable alternative is to use swam algorithms which can obtain approximate solutions to real-world intractable problems. This paper presents two swarm algorithms embodying the adaptive resource bound technique for conquering the constrained nonlinear resource allocation problem. Experimental results manifest that the proposed methods are more effective and efficient than a genetic algorithm-based approach. The convergence behavior of the proposed methods is analyzed by observing the variations of population entropy. Finally, a worst-case analysis is conducted to provide a reliable performance guarantee.**

Keywords-particle swarm optimization; ant colony optimization; metaheuristic; resource allocation

I. INTRODUCTION

Resource management is the central process to ensure the organization's competence against its rivals. It deals with classification, allocation, stocking, processing, storage, and valuation for an organization's resources when they are needed. Among these tasks, *resource allocation problem* (RAP) has been targeted by many researchers due to its everlasting importance. The constrained RAP seeks for an optimal allocation of resources to a number of activities for optimizing organization's objective subject to operational constraints. For instance, project budgeting [1] allocates a given amount of money to a number of projects for maximizing the net present value (NPV) or internal rate of return (IRR), software testing [2] allocates a number of programmers with varying skills to achieve maximum reliability of the software, task allocation [3] allocates a given number of program modules to a number of processors for minimizing the incurred cost, health care financing [4] allocates a fixed amount of medical resource across competing programs promising improved health for patients. For a comprehensive survey the reader is referred to [5].

There exist many solution methods for tackling distinct versions of RAP. We briefly summarize them as follows. (1) Integer linear programming [6] and mixed integer linear programming [4] have been used to formulate and solve the linear RAP with discrete or continuous resource. (2) Basso and Peccati [1] proposed a dynamic programming (DP) algorithm with an efficient pruning procedure for solving the portfolio optimization problem in project financing. Morales et al. [7] presented three parallel DP algorithms using pipeline, dominancy, and resource parallelism to conquer the curse of dimensionality. (3) Bretthauer and Shetty [8] solved the RAP subproblems using a branch-and-bound tree via a one-dimension search for the optimal Lagrange multiplier of the constraint. Bretthauer and Shetty [9] further improved the algorithm by incorporating the pegging method, which iteratively reduces the size of the relaxed subproblems containing variables not satisfying their bounds, for solving the problem more efficiently. (4) Since exact algorithms could be very time expensive for large-scaled problems, an alternative for solving the RAP is to find approximate solutions with reasonable computational time. Dai et al. [2] proposed a genetic algorithm for allocation of software testing resource. The chromosome is represented by a list of modular testing times to be allocated and the objective is to maximize the system reliability with the minimum testing cost. Hou and Chang [10] presented a genetic algorithm for allocating a number of products among plants such that the incurred production cost is minimized.

The motivations of this research are two-fold. *First*, we observe that most of existing methods for tackling the RAP are based on mathematical programming techniques which may fail to derive exact solutions with reasonable time for problems of large size. For the RAP practitioners, they would like to obtain a feasible and quality, although not optimal, solution when the problem size is large. An alternative for obtaining approximate solutions is using metaheuristic algorithms. *Second*, the development of metaheuristic computation has been flourishing during the last decade. Many metaheuristic paradigms such as genetic algorithm (GA) [11], simulated annealing (SA) [12], tabu search (TS) [13], ant colony optimization (ACO) [14], and particle swarm optimization (PSO) [15] have been applied to solve benchmark NP-hard problems. Encouraged by their successful applications, we investigate the feasibility of using metaheuristic algorithms for solving the RAP.

 ISBN: 978-1-61208-154-0

The remainder of this paper is organized as follows. Section 2 formulates the addressed RAP problem. Section 3 presents the proposed swarm algorithms with adaptive resource bounds for tackling the problem. Section 4 reports the comparative performances, convergence analysis, and worst-case analysis. Finally, we conclude in Section 5.

II. Problem Formulation

Given Q units of discrete resource and T activities, the problem is to find an optimal resource allocation to these activities such that the total costs entailed by performing the activities are minimized. The quantity of resource allocated to activity i is constrained in the range $[a_i, b_i]$. The cost function $f_i(x_i)$ is an integer nonlinear function which is dependent upon the quantity x_i of resource the activity i consumes. Formally, the RAP problem considered in this sequel is formulated by the following integer program.

$$\text{Min} \quad J(\boldsymbol{X}) = \sum_{i=1}^{T} f_i(x_i), \tag{1}$$

$$\text{subject to} \quad \sum_{i=1}^{T} x_i = Q, \tag{2}$$

$$0 \le a_i \le x_i \le b_i \le Q;\ \forall i = 1,2,...,T, \tag{3}$$

$$x_i \in \textit{integer}\ .$$

The objective function (1) represents the total cost resulted from the resource allocation decision $\boldsymbol{X} = \{x_i\}_{1 \le i \le T}$. Constraint (2) guarantees that the sum of allotted resource is equivalent to the total units of available resource. Constraint (3) states that the quantity x_i of resource allotted to activity i is constrained between the lower bound a_i and upper bound b_i.

III. Proposed Method

A. Adaptive Resource Bounds

As seen from Constraint (2) of the problem formulation, the quantities of resource allotted to different activities are correlated because all units of resource should be exactly used out during the execution of activities. If we allocate too many units of resource to a certain activity, the remaining resource may be insufficient for performing the other activities. On contrary, if we allocate too few quantities of resource to a certain activity, the amount of remaining resource may be larger than what can be maximally consumed by the other activities. Therefore, exploring all possible resource allocations between the pre-specified fixed resource bounds will yield many infeasible solutions violating Constraint (2) and impair the efficiency of the algorithm.

Two remedies to the above problem are presented as follows. The *feasibility checking rule* checks whether there exist any feasible solutions for the given problem instance. There exist no feasible solutions to the underlying problem if either $\sum_{i=1}^{T} a_i > Q$ or $\sum_{i=1}^{T} b_i < Q$ holds since Constraint (2) can never be satisfied in these conditions. If the given problem instance passes the feasibility checking rule, the *adaptive resource bound updating rule* guides the algorithm to construct a feasible solution. It sequentially allocates resource to the activity and adapts the resource bounds for the next activity. Let Q' be the remaining units of resource (initially $Q' = Q$) and assume that we have allocated resource to the first i activities. We adapts the resource upper bound for the (i+1)th activity as follows.

$$b'_{i+1} = \min\left\{\left(Q' - \sum_{l=i+2}^{T} a_l\right),\ b_{i+1}\right\},\quad i = 0, 1, \ldots, T\text{-}2. \tag{4}$$

The updating for the above resource upper bound is because the (i+1)th activity can only consume at most $Q' - \sum_{l=i+2}^{T} a_l$ for satisfying the minimum resource requests of the rest of the activities, and it cannot exceed the original resource upper bound b_{i+1} of the (i+1)th activity neither. Similarly, the resource lower bound for the (i+1)th activity is updated by

$$a'_{i+1} = \max\left\{\left(Q' - \sum_{l=i+2}^{T} b_l\right),\ a_{i+1}\right\},\quad i = 0, 1, \ldots, T\text{-}2. \tag{5}$$

After allocating resource to the first T - 1 activities with respect to the adaptive resource bounds, the remaining resource should be entirely given to the last activity in order to ensure that all units of the resource have been completely consumed.

B. PSO-based Method

The PSO [15] is inspired by the observations for bird flocking and fish schooling. A number of birds/fish flock synchronously, change direction suddenly, and scatter and regroup together. Each individual, called a particle, benefits from the personal best experience of its own (*pbest*) and that of any members of the swarm (*gbest*) observed so far during the search for food. In what follows, we apply the PSO with the adaptive resource bounds for solving the nonlinear RAP.

Each particle position P_i corresponds to a feasible resource allocation satisfying all constraints. Formally, $P_i = (p_{i1}, p_{i2}, ..., p_{iT})$ where $a'_j \le p_{ij} \le b'_j,\ \forall j = 1, 2, ..., T-1$ and $p_{iT} = Q - \sum_{j=1}^{T-1} p_{ij}$. The particle representation requests an allocation which allots p_{ij} units of resource to the jth activity and the last activity consumes the remaining resource. The PSO initializes a swarm of particles at random. In each iteration, particle i adjusts its velocity v_{ij} and position p_{ij} through each activity dimension j by referring to the personal best position ($pbest_{ij}$) and the swarm's best position ($gbest_j$) as follows.

 ISBN: 978-1-61208-154-0

$$v_{ij} = K[v_{ij} + c_1 r_1(pbest_{ij} - p_{ij}) + c_2 r_2(gbest_j - p_{ij})] \quad (6)$$

and

$$p_{ij} = p_{ij} + v_{ij} \quad (7)$$

where c_1 and c_2 are the self- and socio-cognition coefficients, r_1 and r_2 are random real numbers drawn from $U(0, 1)$, and K is the constriction factor. Clerc and Kennedy [16] has shown that the use of a constriction factor is needed to insure convergence of the PSO, and it is determined by

$$K = \frac{2}{\left|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}\right|} \quad (8)$$

where $\varphi = c_1 + c_2$, $\varphi > 4$. Typically, φ is set to 4.1 and K is thus 0.729.

However, the fly of the particles is constrained by the adaptive resource bounds. We set particle position p_{ij} to a'_j if it passes over the adaptive lower bound, and to b'_j if it exceeds the adaptive upper bound.

The swarm intelligence of the PSO is managed by *pbest* and *gbest*. To identify them, the particle *fitness* should be evaluated. We define the fitness function of particle P_i by $fitenss(P_i) = 1/J(P_i)$, where $J(P_i)$ is computed by the objection function (1) using P_i as the decision for the resource allocation. Thus, the smaller the total cost incurred by P_i, the higher the fitness is, and vice versa. Here we propose a *bounding criterion* to expedite the computation for the fitness. We observe that the fitness value of a particle is only used for determination of $pbest_i$ and *gbest*, but not directly used for velocity update. Since $J(P_i)$ is a monotonically increasing function, we can use the fitness of the incumbent $pbest_i$ as a fitness bound (which should not be confused with the resource bound) and terminate the fitness evaluation of the ith particle when the intermediate fitness value has exceeded this bound. Also, only those $pbest_i$ that have been updated in the current iteration need to be compared with *gbest* for its possible updating. The use of bounding criterion can save the computational time significantly.

In all of the experiments, we terminate the program when our algorithm has experienced a specified number of computations for the fitness function.

C. ACO-based Method

The ant colony optimization (ACO) [14] is inspired by the research on the real ant foraging behavior. Ethologists observed that ants can construct the shortest path from their colony to the feeding source through the use of pheromone trails. An ant leaves some quantities of pheromone on the ground and marks the path by a trail of this substance. The next ant will sense the pheromone laid on different paths and choose one with a probability proportional to the amount of pheromone on it. The ant then traverses the chosen path and leaves its own pheromone. This is a positive feedback process which favors the path along which more ants previously traversed.

To apply ACO for solving the RAP, the problem must be represented by an appropriate graph along which the ant moves to construct candidate solutions. We represent an RAP instance by a (T+2)-layered graph. The first layer (start) and the last layer (sink) consist of only one node with which the ant starts and terminates its traversal. The intermediate T layers represent the allowable resource allocation quantities for the T tasks, respectively. Therefore, for the ith layer the allowable resource allocation quantity ranges from a'_i to b'_i. An ant constructs a candidate solution by traversing a path which starts from the start-node, visits each layer sequentially by selecting a node with an allowable quantity, and finally terminates at the sink-node.

The node transition rule determines the probability which is dependent on two factors: *pheromone* and *visibility*. Without loss of generality, assume that the ant is currently positioned at the jth node of the ith layer and is going to select a node, say, the kth node, from the $(i+1)$th layer. We use pheromone τ_{ijk} to exploit the historical traversal experiences for all the ants and determines the desirability for traversing the edge from a global optimization view. On the other hand, the visibility $\eta_{i+1,k}$ has nothing to do with the ant traversal experiences and only considers the problem-specific static information, which corresponds to the greediness about selecting a node from a local heuristic view. In particular, the visibility value for selecting the kth node from the $(i+1)$th layer is defined as

$$\eta_{i+1,k} = \frac{s}{f_{i+1}(k)}, \; i = 0, 1, \ldots, T\text{-}2; k = a_{i+1}, \ldots, b_{i+1}, \quad (9)$$

where s is a small constant. That is $\eta_{i+1,k}$ grows inversely proportional to $f_{i+1}(k)$. The smaller the cost $f_{i+1}(k)$ incurred by allocating k quantities of resource to the $(i+1)$th task, the higher the value of $\eta_{i+1,k}$.

We define the node transition probability p_{ijk} with which the ant at the jth node of the ith layer moves to the kth node of the $(i+1)$th layer as follows.

$$p_{ijk} = \frac{\tau_{ijk}^{\alpha} \eta_{i+1,k}^{\beta}}{\sum_{l=a'_{i+1}}^{b'_{i+1}} \tau_{ijl}^{\alpha} \eta_{i+1,l}^{\beta}} \quad (10)$$

where α and β are ACO parameters controlling the relative importance ratio between τ_{ijk} and $\eta_{i+1,k}$. Before activating the next iteration, the quantity of pheromone on each edge is updated by the following pheromone updating rule,

 ISBN: 978-1-61208-154-0

$$\tau_{ijk} \leftarrow (1-\rho)\tau_{ijk} + \sum_{t=1}^{m} \Delta\tau^t \tag{11}$$

where $\rho \in (0,1)$ is the evaporation rate of pheromone trails, m is the size of ant population, $\Delta\tau^t$ is calculated by

$$\Delta\tau^t = \frac{s}{J(X_t)} \tag{12}$$

$J(X_t)$ is computed by the objection function (1) using X_t as the decision for the resource allocation.

IV. Experimental Results

In this section, we analyze the properties of the proposed swarm algorithms and present the comparative performances with competing algorithms. The platform for conducting the experiments is a PC with a 2.4 GHz CPU and 256 MB RAM. All programs are coded in C++ language.

A. Synergism

An interesting property about swarm algorithms is whether there exists an optimal size for the swarm. Given a fixed computation resource, the larger the swarm size, the smaller the number of evolutionary iterations can be performed for each particle/ant, and vice versa. We fix the number of times for evaluating the objective function when performing our algorithm with swarms of different size. As shown in Fig. 1, the average cost is minimal when the swarm size equals 20 for both the PSO- and ACO-based algorithms. In other words, there does exist an optimal swarm size for the given problem instance such that the synergism among different individuals is maximized.

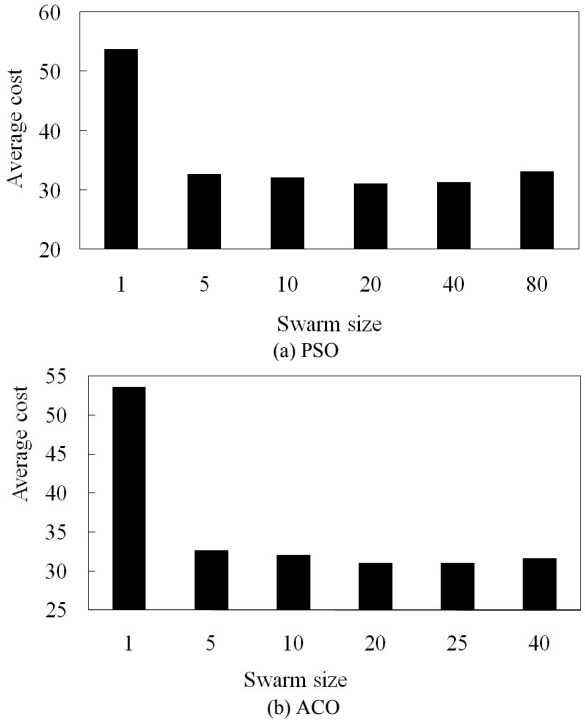


Fig. 1 Average costs obtained by using swarms of different size.

B. Convergence

The convergence of the swarm algorithms can be analyzed by deriving the information entropy contained in the swarm. Here, we propose the information *entropy* for measuring the convergence of *pbest* in PSO and of node transition probability in ACO, because the evolution trajectories of PSO and ACO are substantially drawn by the value of *pbest* and the node transition probability. Fig. 2 shows a typical run for the variations of entropy value as the number of evolutionary iterations increases. It is observed for both swarm algorithms that the entropy value drops drastically at the initial period, then decreases gradually and finally stagnates. This demonstrates that the swarm intelligence is greatly enriched during the initial stage where the swarm scatters in the entire search space; but as the evolution becomes mature, the distributed awareness is resorting to the centralized awareness.

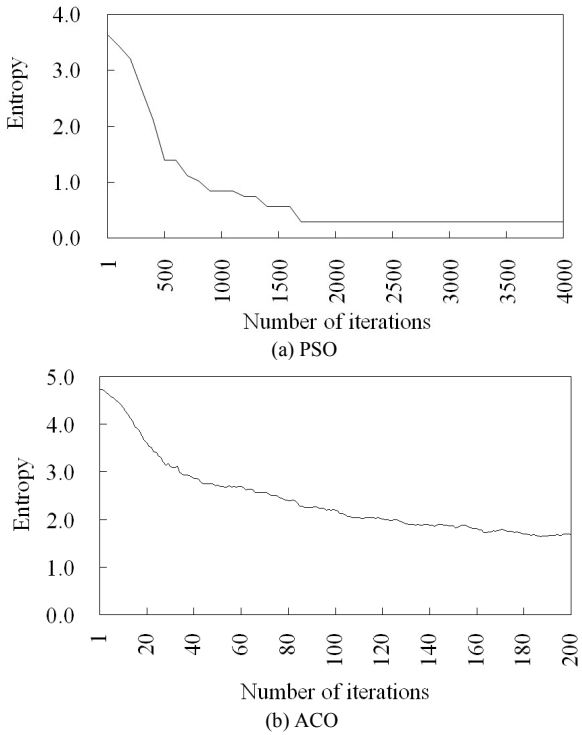


Fig. 2 The entropy as the number of iterations increases.

C. Comparative Performances

The performance of the proposed swarm algorithms is evaluated by competing with an existing GA-based approach [10] and a naïve exhaustive search. We report the average cost and the needed mean CPU time over five independent

TABLE I
COMPARATIVE PERFORMANCES

Q	T	Δ	PSO		ACO		GA		Exhaustive	
			cost	time	cost	time	cost	time	cost	time
100	5	10	17	1301	17.8	48	17.8	3078	17	0.001
	5	20	12	1582	12	152	13.0	3982	12	0.001
	5	30	5	2012	5	326	10.5	4556	5	0.001
200	10	10	26	2954	26	234	38.0	7439	26	0.001
	10	20	12	3565	12	440	25.3	21317	12	8812
	10	30	13.5	3865	13.4	518	29.0	27132	13	142294
300	15	10	32.4	4085	32	416	47.0	194642	32	76159
	15	20	21.6	4927	22	604	60.0	285676	-	> 2 days
	15	30	17.2	6609	17	1007	-	> 2 days	-	-
400	20	10	40.6	5938	40	610	-	-	-	-
	20	20	28.1	7210	28.2	1047	-	-	-	-
	20	30	31.2	9927	31.2	2327	-	-	-	-

runs of each algorithm for every problem instance, while the exhaustive search is executed one time for deriving the exact solution. For practical concerns, all competing algorithms are terminated if the execution time exceeds 48 hours, and we discard such cases for further comparison.

We generate a set of testing problems with diverse computational complexity by varying the number of activities (T) and the mean allowable range of allocated resource units ($\Delta = \sum_{i=1}^{T} (b_i - a_i) / T$) for all activities. The value of T ranges from 5 to 20, and for each value of T, we set Δ to 10, 20, and 30, respectively. The value of Q is given appropriately for various values of T and Δ in order to assure existence of feasible solutions. It is seen from Table 1 that, for the small- and median-sized problems, both PSO and ACO can derive solutions that are equal or very close to the exact solutions as reported by exhaustive search, while the solutions obtained by the GA-based approach are far from the optima. As for the large-sized problems, the solutions obtained by using the PSO and ACO are also significantly better than those obtained by the GA-based approach. This is mainly due to the adaptive resource bounds technique for avoiding producing infeasible solutions and thus significantly reducing the searching space. On the other hand, the GA-based method does not actually reduce the searching space, when infeasible solutions are produced it modifies them to feasible solutions by adjusting resource allocation between activities to ensure that the resource allocation does not violate resource bound constraints.

As for the computational efficiency, Table 1 also illustrates that the CPU times consumed by the exhaustive search method grow exponentially with the problem size, and it fails to solve large-scaled problems. The computational times consumed by GA also grow at a fast rate and exceed 48 hours for the last four instances. On the other hand, the computational time used by the PSO and ACO methods as the problem size increases is still in an acceptable range, thought the ACO-based method seems to be computationally faster than the PSO-based method. In summary, both the proposed swarm algorithms can derive quality solutions against scalable problems.

D. Worst-case Analysis

Since swarm algorithms are stochastic methods, each separate run of the same program could yield different result. It is critical to analyze the worst-case performance one may obtain if the swarm algorithm is adopted for tackling the RAP. The worst-case performance is analyzed as follows. *First*, the program is executed for a specific number of repetitive runs and each run will output an optimal solution. *Second*, the worst-case analysis is set up as the worst solution we could get from those optimal solutions. Fig. 3 shows the worst-case analysis where the swarm algorithms are executed to solve the problem instance with (Q, T, Δ) = (400, 20, 30) for 1,000 times. The curve shows that about 97% of the repetitive runs can obtain a quality solution with cost less than or equal to 33, meaning that we can be confident in using the swarm algorithms for solving the RAP. When the user requests a worst-case guarantee for the derived solution with cost no more than 33, he/she can obtain such a solution by executing the swarm algorithms for no more than 30 repetitive runs (3% of 1,000 runs).

V. CONCLUSIONS

The constrained resource allocation problem (RAP) seeks for an allocation of a fixed amount of resource to a number of activities such that the objective is optimized and the resource constraints are met. RAP has many applications, including product allocation, resource distribution, project budgeting, software testing, health care resource allocation, just to name a few. Different versions of problem formulations have been proposed in accordance with various applications. This paper addressed the nonlinear RAP with integer decision variable constraint. The proposed swarm algorithms are built on the foundation of PSO and ACO with the adaptive resource bounds technique in order to construct feasible solutions. Simulation results show that the swarm algorithms derive comparable solutions to the exact solutions obtained using an exhaustive search method. The swarm algorithms also outperform a GA-based approach which yields worse solutions and needs more computational time. The general

 ISBN: 978-1-61208-154-0

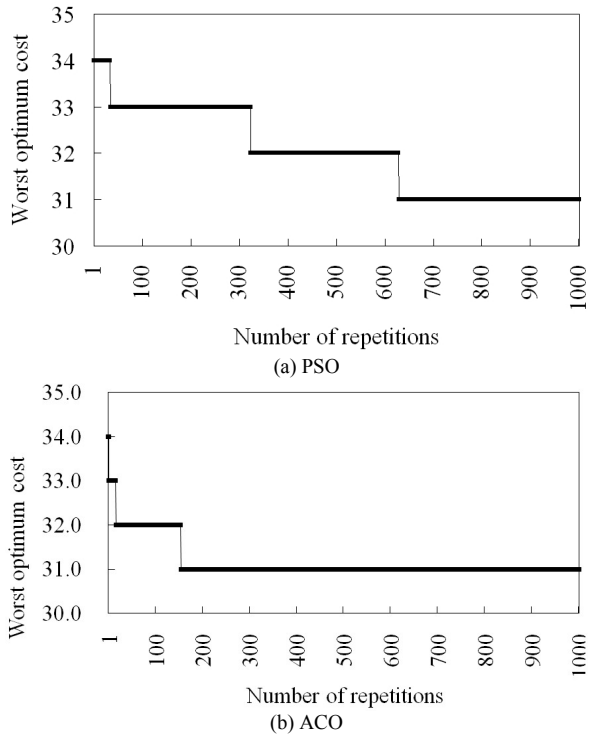


Fig. 3 The worst optimum cost vs. the number of repetitive runs of the PSO algorithm.

properties and the convergence behaviors of the swarm algorithms are analyzed. Finally, a performance guarantee is provided through the worst-case analysis.

ACKNOWLEDGMENT

This work is supported in part by the National Science Council of Taiwan under grant NSC 98-2410-H-260-018-MY3.

REFERENCES

[1] A. Basso and L. A. Peccati, Optimal resource allocation with minimum activation levels and fixed costs, *European Journal of Operational Research*, vol. 131, 2001, pp. 536-549.

[2] Y. S. Dai, M. Xie, K. L. Poh, and B. Yang, Optimal testing –resource allocation with genetic algorithm for modular software systems, *The Journal of Systems and Software*, vol. 66, 2003, pp. 47-55.

[3] A. Ernst, H. Hiang and M. Krishnamoorthy, Mathematical programming approaches for solving task allocation problems, *Proc. of the 16th National Conf. Of Australian Society of Operations Research*, 2001.

[4] A. A. Stinnett and A. D. Paltiel, Mathematical programming for the efficient allocation of health care resource, *Journal of Health Economics*, vol. 15, 1996, pp. 641-653.

[5] T. Ibaraki and N. Katoh, "Resource allocation problems: algorithmic approaches," MIT Press, Boston, 1988.

[6] S. Birch and A. Gafni, Cost effectiveness analysis: Do current decision rules lead us where we want to be?, *Journal of Health Economics*, vol. 11, 1992, pp. 279-296.

[7] D. Morales, F. Almeida, F. Garcia, J. L. Roda, and C. Rodriguez, Design of parallel algorithms for the single resource allocation problem, *European Journal of Operational Research*, vol. 126, 2000, pp. 166-174.

[8] K. M. Bretthauer and B. Shetty, The nonlinear resource allocation problem, *Operations Research*, vol. 43, 1995, pp. 670-683.

[9] K. M. Bretthauer and B. Shetty, A pegging algorithm for the nonlinear resource allocation problem, *Computers & Operations Research*, vol. 29, 2002, pp. 505-527.

[10] Y. C. Hou and Y. H. Chang , A new efficient encoding mode of genetic algorithms for the generalized plant allocation problem, *Journal of Information Science and Engineering*, vol. 20, 2004, pp. 1019-1034.

[11] D. E. Goldberg, "Genetic algorithms in search, optimization, and machine learning," Addison Wesley, Reading, Massachusetts, 1997.

[12] S. Kirkpatrick, C. Gelatt Jr., and M. Vecchi, Optimization by simulated annealing, *Science*, vol. 220, 1983, pp. 671-680.

[13] F. Glover, Tabu search - Part I, ORSA *Journal of Computing*, vol. 1, 1989, pp. 190-206.

[14] M. Dorigo, "Optimization, learning, and natural algorithms," Ph.D. Thesis, Dip. Elettronica e Informazione, Politecnico di Milano, Italy, 1992.

[15] J. Kennedy and R.C. Eberhart, Particle swarm optimization, *Proceedings IEEE Int'l. Conf. on Neural Networks*, IV, 1995, pp. 1942-1948.

[16] M. Clerc and J. Kennedy, The particle swarm explosion, stability, and convergence in a multidimensional complex space, *IEEE Transaction on Evolutionary Computation*, vol. 6, 2002, pp. 58-73.

 ISBN: 978-1-61208-154-0

A Multi-Answer Character Recognition Method and Its Implementation on a High-Performance Computing Cluster

Qing Wu, Morgan Bishop, Robinson Pino,
Richard Linderman
Information Directorate
US Air Force Research Laboratory
Rome, New York, USA
{Qing.Wu, Morgan.Bishop, Robinson.Pino,
Richard.Linderman}@rl.af.mil

Qinru Qiu
Department of Electrical Engineering & Computer Science
Syracuse University
Syracuse, New York, USA
qinru.qiu@gmail.com

***Abstract* — In this paper, we present our work in the implementation and performance optimization of a novel multi-answer character recognition method on a high-performance computing cluster. The main algorithm used in this method is called the Brain-State-in-a-Box (BSB), which is an auto-associative neural network model. We applied optimization techniques on different parts of the BSB algorithm to improve the overall computing and communication performance of the system. Furthermore, the proposed method adopts a new way to train, recall, and organize the BSB models for different characters, in order to provide a sorted (based on recall convergence speed) list of candidates for a given character image.**

Keywords – character recognition, brain-state-in-a-box, neural network, performance optimization

I. INTRODUCTION

In the past four decades, a significant amount of research work [7][8] has been performed on optical character recognition (OCR) for printed characters. OCR for printed text represents an important application of pattern recognition and image processing. As of today, there are many software OCR tools available, such as the open-source Tesseract-OCR [9].

Although OCR for printed text is regarded as a largely solved problem, its reliability and robustness are not guaranteed when the input text image is either noisy or severely occluded. The main limitation of the existing OCR tools is that they are trying to provide a single answer for each individual character. When a character image is severely occluded, most OCR algorithms will make a "best guess" and give one deterministic answer, which does not provide a "second-thought" candidate. On the other hand, the human cognition process looks at not only the individual character image, but also its context such the word and the sentence. When a character is hard to recognize, a human will first make multiple guesses and cross-reference them with context, then decide which guess fits the word best, which word fits the sentence best, etc.

The goal of our research at AFRL is to develop a high-performance text recognition software tool that is highly reliable and robust for noisy or occluded text images. The multi-answer character recognition method introduced in this paper is an essential component of the project. Other major components of the project include a word-level language model and a sentence-level language model [12], which are not in the scope of this paper. The overall software architecture makes these neuromorphic algorithms work together to produce improved text recognition results.

The text recognition software is implemented on a high-performance computing (HPC) cluster that consists of almost 70,000 processor cores and provides a massive peak computing power of 500 trillion floating-point operations per second. To take full advantage of the available HPC resources, the algorithm must be highly scalable so that the overall performance increases linearly with the number of processor cores used. Furthermore, HPC resources are most effective when performing regular and continuous floating-point operations. Through optimization efforts, we found the Brain-State-in-a-Box (BSB) neural network model [1][2][3] to fit the HPC efficiently.

In order to provide multiple answers to a given character image, we designed a new "racing" mechanism when performing pattern recognition using the BSB models. Basically, through a training process we build different BSB models for different characters. Any input character image will be sent to all the BSB models for recall (pattern recognition). When all recalls are completed, a set of candidates is selected based on the convergence speed. This process forms the proposed multi-answer character recognition method.

The remainder of the paper is organized as follows. In Section II we provide background information on the system architecture, the BSB model, and processor architecture. Section III describes the details of optimizing the BSB algorithm on the IBM Cell-BE processor. Section IV discusses the implementation of the "racing" mechanism to generate multiple recognition candidates, as well as the simulation results.

II. BACKGROUND

A. Massively Parallel Computing System

Modeling and simulation of human cognizance functions involve large-scale mathematical models, which demand a

 ISBN: 978-1-61208-154-0

high performance computing platform. We desire a computing architecture that meets the computational capacity and communication bandwidth of a large-scale associative neural memory model. In particular, we are interested in processing pages of text at real-time rates of up to 50 pages per second.

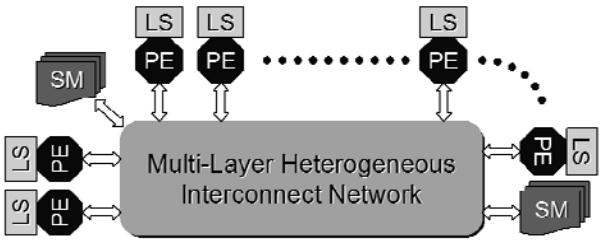


Figure 1. High performance cognitive computing system.

Figure 1 shows the targeted architecture of a high performance cognitive computing system. This system consists of multiple Processing Elements (PEs) interconnected with a multi-layer heterogeneous interconnect network. All PEs have access to a local memory called the Local Store (LS) and Shared Memory (SM) connected to the interconnect network.

B. Brain-State-in-a-Box Model

The BSB model is a simple, auto-associative, nonlinear, energy-minimizing neural network [1][2][3]. A common application of the BSB model is to recognize a pattern from a given occluded version. It can also be used as a pattern recognizer that employs a smooth nearness measure and generates smooth decision boundaries3.

There are two main operations in a BSB model, Training and Recall. In this paper, we will focus on the BSB recall operation. The mathematical model of a BSB recall operation can be represented in the following form:

$$\mathbf{x}(t+1) = S(\alpha * \mathbf{A} * \mathbf{x}(t) + \lambda * \mathbf{x}(t) + \gamma * \mathbf{x}(0)) \qquad (1)$$

where:

- **x** is an *N* dimensional real vector
- **A** is an *N*-by-*N* connection matrix
- **A*****x** *(t)* is a matrix-vector multiplication operation
- α is a scalar constant feedback factor
- λ is an inhibition decay constant
- γ is a nonzero constant if there is a need to maintain the input stimulation

S () *is the "squash" function defined as follows:*

$$S(y) = \begin{cases} 1 & if \quad y \geq 1 \\ y & if \quad -1 < y < 1 \\ -1 & if \quad y \leq -1 \end{cases} \qquad (2)$$

Note that in the proposed algorithm, we choose λ to be 1.0 and γ to be 0.0. But they can be easily changed to other values during the implementation. Given an input pattern **x**(0), the recall process computes Equation (1) iteratively to reach convergence. A recall converges when all entries of **x**(*t*+1) are either "1.0" or "-1.0". In our implementation, it usually takes more than ten iterations for recall to converge.

C. Cell Broadband Engine Architecture

The IBM Cell Broadband Engine (Cell-BE) architecture [4][5][6] is a multi-core architecture (shown in Figure 2) designed for high performance computing. The architecture features nine microprocessors on single chip. A Power architecture compliant core called the *Power Processing Element* (PPE) and eight other attached processing cores called *Synergetic Processing Elements* (SPEs) are interconnected by a high bandwidth *Element Interconnect Bus* (EIB). This heterogeneous architecture with high performance computing cores is designed for distributed multicore computing.

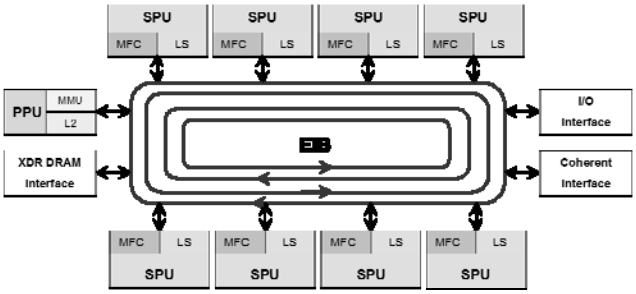


Figure 2. Cell-BE processor block diagram.

III. Performance Optimization of the BSB Algorithm on the Cell-BE Processor

In this section we will describe the implementation and performance optimization of the recall operation on a single Cell-BE processor. One of the major challenges in implementing the recall operation in software is the high computational demand. For one 128-dimensional BSB model recall we need a matrix-to-vector multiplication that involves 16384 floating-point multiplications and 16256 floating-point additions. In addition, we also need 128 floating-point multiplications for the feedback factor and 256 comparisons. Our final large-scale associative neural model would involve a large number of BSB models (up to 500,000). We need a parallel distributed computational model which can perform a massive number of BSB recall operations in parallel.

The Cell-BE processor with high performance computing cores is an ideal platform for distributed computing. We can run one BSB recall on each SPE. Next, we will describe the implementation details of a 128-dimensional BSB recall operation on one SPE.

A. Matrix-Vector Multiplication

Multiplication of a 128x128 matrix with a 128x1 vector can be represented as follows. We are showing these rather

 ISBN: 978-1-61208-154-0

simple operations in detail, in order to elaborate the floating-point operations involved in the computation.

$ax_0 = a_{0,0}*x_0 + a_{0,1}*x_1 + a_{0,2}*x_2 + \ldots + a_{0,127}*x_{127}$
$ax_1 = a_{1,0}*x_0 + a_{1,1}*x_1 + a_{1,2}*x_2 + \ldots + a_{1,127}*x_{127}$
$ax_2 = a_{2,0}*x_0 + a_{2,1}*x_1 + a_{2,2}*x_2 + \ldots + a_{2,127}*x_{127}$
..
..
$ax_{127} = a_{127,0}*x_0 + a_{127,1}*x_1 + a_{127,2}*x_2 + \ldots + a_{127,127}*x_{127}$

The scalar implementation of the above equations can be written in C as:

```
float ax[128], x[128], a[128*128];
int row,col;

for(row=0;row<128;row++){
   ax[row]=0;
   for(col=0;col<128;col++) {
        ax[row]+=x[col]*a[row*128+col];  }}
```

We can improve the computational performance by using the Single Instruction Multiple Data (SIMD) model of the SPE. Most of the instructions in SPEs operate on 16 bytes of data and also the data fetching from local store is 16-byte aligned. To implement the scalar computation using SIMD instructions, the compiler has to keep track of the relative offsets between the scalar operands to get the correct results. This implementation ends up having additional rotation instructions added, which is an overhead. To get better performance and the correct result, it is wise to handle the data as vectors of 16 bytes (or four single-precision floating-point numbers) each.

The matrix multiplication using SIMD instructions can be implemented as below.

```
float ax[128] __attribute__((aligned (128)));
float x[128] __attribute__((aligned (128)));
float a[128*128] __attribute__((aligned (128)));
int row,col;
vector float  *x_v, *a_v;
vector float temp;
x_v = (vector float *)x;
a_v = (vector float *)a;

for(row=0;row<128;row++){
  temp = (vector float){0.0,0.0,0.0,0.0};
  for(col=0;col<128/4;col++) {
    temp=spu_madd(x_v[col],a_v[row*32+col],temp);
  }
  ax[row] =
(spu_extract(temp,0)+spu_extract(temp,1)+
 spu_extract(temp,2)+spu_extract(temp,3));
}
```

spu_madd and *spu_extract* are intrinsics which make the underlying Instruction Set Architecture (ISA) and SPE hardware accessible from the C programming language. *spu_madd* is the C representation for the multiply and add instruction. *spu_extract* returns the vector member value specified by the offset. Compared to the scalar implementation, SIMD code reduces 16384 multiplication operations to 4096 vector multiplications. The above implementation still performs scalar addition to get the final result. We can further improve the performance by rearranging the matrix so that we can apply SIMD instructions on all the matrix-vector multiplication operations.

We divide the entire matrix into smaller 4x4 matrices. Elements of each of these 4x4 matrices are shuffled according to a specific pattern as shown in Figure 3.

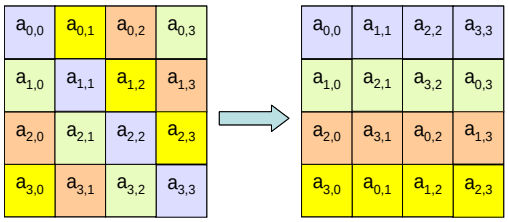


Figure 3. Matrix shuffling from original order to SIMD order.

The shuffled matrix is multiplied with the X vector as shown in Figure 4. Note that each row of the shuffled matrix is a vector of four single-precision floating-point numbers. And (x_0, x_1, x_2, x_3) is the other vector in the SIMD operation.

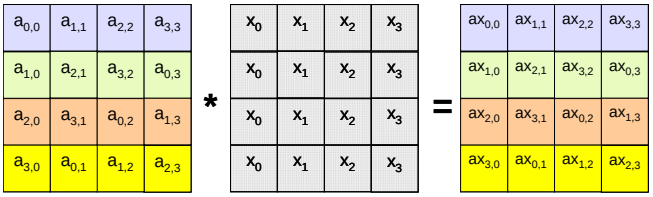


Figure 4. Shuffled matrix multiplication.

To obtain the final result we need to rotate the some of the elements to align them back to their original offset and add all the rows, as shown in Figure 5.

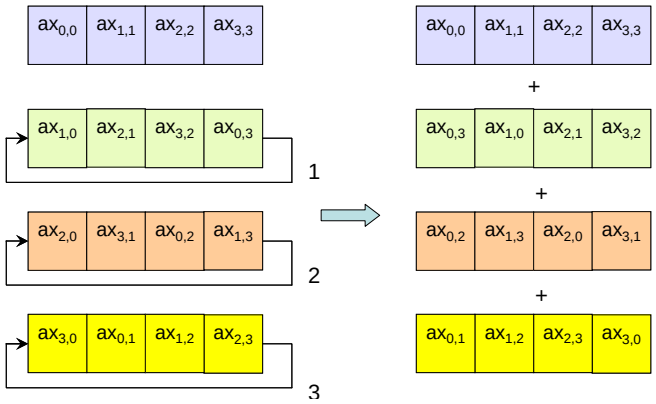


Figure 5. Product alignment and accumulation.

By shuffling the input matrix we have effectively replaced the 3*128 scalar additions from the previous implementation with 3*32 rotations and also we reduced the overhead added by the compiler to perform scalar addition.

For every four rows of the 128x128 matrix, we will have 32 4x4 matrices. The above shuffle-multiply-rotate-accumulate operation is repeated for each one of them, and

 ISBN: 978-1-61208-154-0

at the end we will be able to obtain the four elements of the resulting 128x1 vector. For 128 rows, we need to repeat the above procedure 32 times and obtain the complete result. In our current program implemented on the PS3, we assume that the 128x128 matrix has already been shuffled before being stored into the main memory.

B. Other operations in BSB recall

From Equation (1) we know that $\mathbf{A}*\mathbf{x}(t)$ has to be multiplied with feedback constant α and the result added with $\mathbf{x}(t)$. This multiplication and addition can be performed by using the *spu_madd* intrinsic, which multiplies two vectors and adds the result to the third vector. This step requires 32 *spu_madd* intrinsics.

The final operation in recall is the squash function. As given in Equation (2) this operation needs two comparisons to check whether the result of the previous computation is >1 or <-1. To perform this kind of compare and assign operation on vector data, the SPE provides special instructions.

spu_cmpgt is an intrinsic which performs element-wise comparisons on two given vectors. If an element in the first vector is greater than corresponding element in the second vector then all the entries of the corresponding element in the result vector are set to '1', else '0'. This result can be used as a multiplexer select: '1' assigns input_1 to the output and '0' assigns input_0 to the output. An intrinsic called *spu_sel* is used to perform the selection. *spu_sel* takes two input vectors and a select pattern. For each entry in the 128-by-1 vector, the corresponding entries from either input vector-0 or input vector-1 are selected.

C. Balancing computation and communication

One of the important factors affecting the performance of the SPE is the data transfer time. Due to limited local store size it is not possible to get all the data required for the computation at once. Therefore the SPE has to initiate direct memory access (DMA) transfers whenever it requires additional data from the main memory, which takes a significant amount of time. However we can leverage the communication-computation concurrency provided by the Cell-BE's asynchronous DMA model by performing computation while data for future computations is being fetched. This is called the *double buffering* method. Figure 6 shows the computational flow with and without double buffering. In the regular communication model, the weight matrix required for the recall is fetched and ten recall iterations are performed. Then the DMA request for the weight matrix of the next recall is initiated. This induces gaps in the computational flow that reduce the effective throughput. In the double buffering implementation, when the computation of the current recall starts, a DMA request for the weight matrix of the next recall is initiated in parallel. The memory controller of the SPE can work in background to fetch the data from the memory. By the time ten iterations of the current recall are completed, the data for the next recall will be available, and then the SPE can continue with its computational flow.

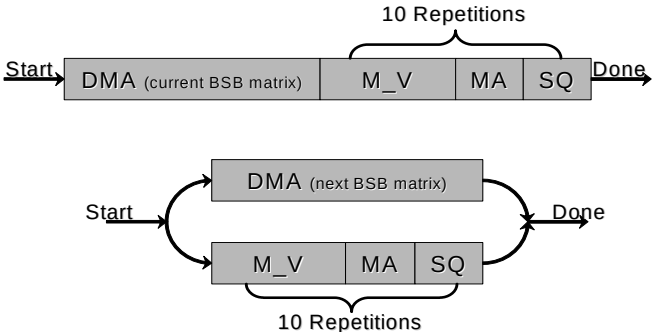


Figure 6. Algorithmic flow without (top) and with (bottom) the double buffering method.

IV. MULTI-ANSWER IMPLEMENTATION AND EXPERIMENTAL RESULTS

In this section, we first describe the "racing" mechanism that we use to implement the multi-answer character recognition process. Then, we present and discuss the experimental results when we apply this method on character images with different amounts of added noise.

A. Multi-answer character recognition using BSB models

Without loss of generality, assume that the set of characters we want to recognize from images consists of 52 characters, which are the upper and lower case characters of the English alphabet.

$$S = \{'a', 'b', \dots, 'z', 'A', 'B', \dots, 'Z'\}$$

We also assume that for each character in *S*, there are *M* typical variations in terms of different fonts, styles, and sizes. For example, the set of images of character '*a*' with different variations can be represented as:

$$S_a = \{a_1, a_2, \dots, a_M\}.$$

In terms of pattern recognition, there are a total of 52**M* patterns to remember during training and to recognize during recall. If we follow the traditional application approaches of the BSB models, the solution is to train one BSB model to remember all the 52**M* patterns. During recall, given an input image, this model will eventually converge to one of the remembered patterns (attractors) that represent the recognition result. The shortcomings of this approach is that firstly it requires a BSB model with large dimensionality (*N* the dimension of vector *X* in Equation 1) to remember all the patterns. This increases the complexity ($\propto N^2$) of the computation and also reduces the scalability when implemented on parallel computing architectures. Secondly, this approach only provides one answer to the input image. The BSB recall process does not return the second or third closest attractor for the image. For recognizing corrupted texts, providing only one answer is not adequate for the low-level pattern recognition model to work with high-level language models.

 ISBN: 978-1-61208-154-0

Therefore, in our implementation, the primary goal is to design a process that provides multiple candidates for an input image. And the secondary goal is to have reasonably-sized BSB models to have good scalability and keep computation complexity under control.

The solution we designed is to use one BSB model for each character in *S*. Therefore there will be a set of 52 256-dimensional BSB models, that is:

$$S_{BSB} = \{BSB_a, BSB_b, \ldots, BSB_z, BSB_A, BSB_B, \ldots, BSB_Z\}.$$

Each BSB model is trained for all variations of a character. For example, BSB_a is trained to remember all the variable patterns in S_a, BSB_b will remember patterns in S_b, so on so forth. If we define the procedure "Recall(*A*, *B*)" as the recall process using model A with input image *B*, which returns the number of iterations it takes to converge, the recall and candidate selection process can be described as follows.

```
Input: character image X.
1. For each trained BSB model BSB_i in S_BSB
        Conv[i] = Recall(BSB_i, X);

2. Sort Conv[i] from low to high to form
   sorted list Conv_s[j];

3. Pick the first K in Conv_s[j] as
   recognition candidates, if it satisfies
   both conditions listed below:

   a. Conv_s[j] <= Th_1;
   // Convergence speed threshold
   b. Conv_s[j] – Conv_s[j-1] <= Th_2;
   // Separation threshold
```

In this algorithm, {*K*, Th_1, Th_2} are adjustable parameters based on overall reliability and robustness needs.

Generally speaking, in our multi-answer implementation, we utilize the BSB model's convergence speed to represent the "closeness" of an input image to the remembered characters (with variations). Then we pick up to *K* "closest" candidates (that satisfy conditions 3a and 3b) to work with high-level language models to determine the final output. In a HPC platform consisting of many (up to 1,700) IBM Cell-BE processors, our implementation was able to execute the recall operations in parallel. Because each BSB model is small enough to fit on a single Cell-BE processor, the overall performance scales linearly with the number of Cell-BE processors used.

B. Simulation results

In our current implementation, the BSB models are trained for 93 characters, with up to six different fonts, five different sizes and two different styles, i.e., up to 60 variations per character. The complete set covers all the characters that can be found on a computer keyboard.

To better demonstrate the general trends of our approach, we will present the results when we use 256-dimensional BSB models trained for the first 52 characters in the set, with two fonts, two sizes and one style (four variations for each character). The input character image is a 15-by-15, 225-pixel grayscale or black-and-white bitmap. Therefore the dimensionality of the BSB models must be at least 225. We chose 256 because it is the next powers-of-two number. In addition, for the given number of the character variations it needs to remember, a 256-dimensional BSB model should have enough attractors.

Table 1 shows the top-three candidates generated by our program when non-occluded character images are given to the models. In the table, the "C" columns show the first, second and third fastest converging BSB models, and the "N" columns show the number of recall iterations before convergence. A "*" mark means that the BSB model has not converged by the limit (Th_1 = 75) to be selected as a candidate. The cells highlighted in the diagonal-line pattern represent the "correct" candidates. We can see that for non-occluded images, the "correct" BSB models always converge first. We also see a 2x to 3x margin in terms of the "N" value between the first and second candidates.

Table 2 and Table 3 show the top-three candidates when the input images are occluded by 1-pixel-strike-through and 3-pixel-strike-through black bars, respectively. Please note the characters shown in the "Input" columns only indicate the damaged characters, but are not the actual images. For a 1(3)-pixel-strike-through, we added one (three) horizontal black bar(s) in the middle of the input image, from the left edge to right edge. From Table 2 we can see that for moderate occlusion, the "correct" BSB models still converge the fastest, although slower as compared to Table 1. However for significant occlusion as in Table 3, we can see that two of them did not converge first, while five of them (highlighted in red) are not among the top-three candidates.

Overall, we have shown that the "closeness" of the input pattern to the remembered ones can be measured by the speed of convergence of the BSB recall process, which we can use to select multiple candidates for an input character image. To deal with the situation when the "correct" BSB is not in the candidate list, we can lower the iteration threshold (Th_1) to, for example, 30. This change will result in "no candidate" for some occluded input, which also means that the high-level language model will receive a candidate list with all characters in it.

Working as a component of the text recognition software developed by AFRL/RI, the absolute accuracy of the proposed character recognition method is not as important as it may be in other character recognition software tools. Our text recognition software has unique language models and algorithms [10][11][12] to work with the BSB outputs. The overall integrated approach achieves better text recognition accuracy, particularly when the character images are damaged.

V. CONCLUSION

We have presented work in the implementation and performance optimization of a novel multi-answer character recognition method on a high-performance computing

 ISBN: 978-1-61208-154-0

cluster. We applied optimization techniques on different parts of the BSB algorithm to improve the overall computing and communication performance of the system. Furthermore, the proposed method adopts a new way in training, recalling, and organizing the BSB models for different characters, in order to provide a list of candidates for a given character image. By offering multiple answers, this character recognition algorithm was able to be integrated with high-level language models to achieve more reliable and robust text recognition capabilities.

ACKNOWLEDGMENT OF SUPPORT AND DISCLAIMER

Received and approved for public release by AFRL on 04/14/2011, case number 88ABW-2011-2178.

The contractor's work is supported by the Air Force Research Laboratory, under contract FA8750-09-2-0155.

Any Opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of AFRL or its contractors.

REFERENCES

[1] J. A. Anderson, J. W. Silverstein, S. A. Ritz, and R. S. Jones, "Distinctive features, categorical perception, probability learning: Some applications of a neural model," Neurocomputing; Foundations of Research, J. A. Anderson and E. Rosenfeld, Editors, The MIT Press, 1989, ch. 22, pp. 283–325, reprint from Psychological Review 1977, vol. 84, pp. 413–451.

[2] M. H. Hassoun, Editor, "Associative Neural Memories: Theory and Implementation," Oxford University Press, 1993.

[3] A. Schultz, "Collective recall via the Brain-State-in-a-Box network," IEEE Transactions on Neural Networks, vol. 4, no. 4, pp. 580–587, July 1993.

[4] T. Chen, R. Raghavan, J. Dale, and E. Iwata, "Cell Broadband Engine Architecture and its first implementation," IBM, 2011, http://www.ibm.com/developerworks/power/library/pa-cellperf/.

[5] "IBM Cell Broadband Engine Architecture," https://www-01.ibm.com/chips/techlib/techlib.nsf/techdocs/1AEEE1270EA2776387257060006E61BA/$file/CBEA_v1.02_11Oct2007_pub.pdf, 2011.

[6] "IBM Cell Broadband Engine resource center," http://www-128.ibm.com/developerworks/power/cell/, 2011.

[7] J. Mantas, "An Overview of Character Recognition Methodologies," Pattern Recognition, 19(6); 425-430, 1986.

[8] G. Nagy, "Optical Character Recognition – Theory and Practice," Handbook of Statistics, Vol. 2, 621-649, 1982.

[9] "Tesseract-OCR", http://code.google.com/p/tesseract-ocr/, 2011.

[10] Q. Qiu, Q. Wu, D. Burns, M. Moore, M. Bishop, R. Pino, and R. Linderman, "Confabulation Based Sentence Completion for Machine Reading," Proceedings of the IEEE Symposium Series on Computational Intelligence, Paris, France, April 2011.

[11] Q. Qiu, Q. Wu, M. Bishop, M. Barnell, R. Pino, and R. Linderman, "An Intelligent Text Recognition System on the AFRL Heterogeneous Condor Computing Cluster," to appear, Proceedings of the DoD High Performance Computing Modernization Program Users Group Conference, Portland, Oregon, June 2011.

[12] Q. Qiu, Q. Wu, and R. Linderman, "Unified Perception-Prediction Model for Context Aware Text Recognition on a Heterogeneous Many-Core Platform," to appear, Proceedings of the International Joint Conference on Neural Networks, San Jose, California, July 2011.

Table 1. Top-three candidates generated when given clean character images.

Input	Top-3 Candidates						Input	Top-3 Candidates					
	1st		2nd		3rd			1st		2nd		3rd	
	C	N	C	N	C	N		C	N	C	N	C	N
a	a	12	I	31	O	31	A	A	12	I	36	z	38
b	b	11	L	28	t	31	B	B	11	z	30	F	30
c	c	12	o	25	e	28	C	C	11	I	27	E	34
d	d	12	I	31	g	32	D	D	12	E	28	f	32
e	e	12	B	35	C	35	E	E	11	F	27	z	29
f	f	11	t	27	E	28	F	F	11	E	22	L	26
g	g	12	I	28	q	30	G	G	11	I	30	D	33
h	h	11	L	26	f	29	H	H	12	L	27	f	28
i	i	11	l	25	I	27	I	I	11	l	22	z	30
j	j	11	v	31	l	33	J	J	12	I	26	g	31
k	k	11	L	26	E	29	K	K	11	L	28	N	28
l	l	11	I	24	z	30	L	L	11	b	29	z	30
m	m	11	f	28	n	28	M	M	11	Y	32	F	35
n	n	12	L	30	b	34	N	N	12	U	26	E	27
o	o	12	c	27	e	29	O	O	11	I	28	G	30
p	p	12	L	28	z	32	P	P	11	L	26	z	32
q	q	11	g	23	I	29	Q	Q	11	I	28	z	32
r	r	12	l	27	I	33	R	R	11	B	30	f	32
s	s	12	I	31	a	32	S	S	11	I	28	C	33
t	t	11	L	25	b	30	T	T	11	l	23	I	23
u	u	12	n	25	D	33	U	U	12	b	24	z	34
v	v	11	I	33	n	36	V	V	11	I	35	n	39
w	w	12	j	35	n	35	W	W	12	f	32	E	32
x	x	11	k	32	n	33	X	X	12	Z	30	v	34
y	y	11	E	31	F	32	Y	Y	11	I	29	R	33
z	z	11	I	29	H	31	Z	Z	11	I	29	n	35

 ISBN: 978-1-61208-154-0

Table 2. Top-three candidates generated when given 1-pixel-strike-through character images.

Input	Top-3 Candidates						Input	Top-3 Candidates					
	1st		2nd		3rd			1st		2nd		3rd	
	C	N	C	N	C	N		C	N	C	N	C	N
a	a	23	R	32	I	33	A	A	23	e	34	z	37
b	b	23	L	28	j	32	B	B	22	F	30	L	30
c	c	24	e	24	o	27	C	C	24	I	30	J	37
d	d	23	g	33	J	34	D	D	23	E	28	f	33
e	e	22	B	35	C	35	E	E	22	F	27	N	29
f	f	23	t	28	E	30	F	F	22	E	25	L	28
g	g	23	q	30	I	31	G	G	23	O	32	H	33
h	h	23	L	28	H	29	H	H	22	f	28	L	29
i	i	23	l	26	I	29	I	I	23	l	24	z	30
j	j	23	a	33	e	33	J	J	23	I	28	R	31
k	k	23	L	29	H	32	K	K	23	L	29	N	29
l	l	23	I	26	z	31	L	L	23	E	27	z	30
m	m	23	f	30	n	31	M	M	23	F	35	U	35
n	n	23	L	30	e	33	N	N	23	F	28	U	29
o	e	24	o	24	c	27	O	O	24	I	31	s	35
p	p	23	L	31	B	33	P	P	23	L	28	r	34
q	q	24	g	25	I	31	Q	Q	23	I	31	R	33
r	r	23	l	29	J	33	R	R	22	f	31	B	33
s	s	22	B	33	a	34	S	S	24	I	30	C	34
t	t	23	L	28	b	33	T	T	23	l	25	I	25
u	u	23	n	28	G	35	U	U	23	b	26	L	34
v	v	24	k	37	B	37	V	V	23	B	40	I	42
w	w	26	j	36	N	38	W	W	28	L	35	f	36
x	x	23	k	35	n	35	X	X	23	Z	34	v	35
y	y	23	H	34	n	36	Y	Y	23	I	34	R	34
z	z	23	I	29	L	34	Z	Z	24	I	29	R	36

Table 3. Top-three candidates generated when given 3-pixel-strike-through character images.

Input	Top-3 Candidates						Input	Top-3 Candidates					
	1st		2nd		3rd			1st		2nd		3rd	
	C	N	C	N	C	N		C	N	C	N	C	N
a	a	26	e	36	B	41	A	A	26	e	36	y	38
b	b	35	t	35	L	39	B	B	28	P	28	L	32
c	c	29	o	36	G	46	C	G	34	e	36	C	37
d	d	31	J	36	K	39	D	D	27	E	32	O	32
e	c	37	G	38	m	45	E	E	29	F	30	L	32
f	f	27	t	33	e	37	F	F	30	f	31	E	31
g	q	36	I	42	J	44	G	G	27	I	43	a	44
h	h	28	H	30	f	33	H	H	29	h	33	i	35
i	l	31	J	36	e	38	I	I	27	i	29	l	29
j	j	30	J	35	K	38	J	J	27	I	35	R	37
k	k	26	L	35	i	37	K	K	26	f	36	i	36
l	l	27	i	29	I	33	L	L	27	E	33	a	37
m	m	26	f	39	n	41	M	M	26	A	44	R	44
n	n	31	o	36	L	39	N	N	26	i	38	E	38
o	o	34	c	35	E	42	O	s	32	G	34	O	34
p	p	26	D	33	e	37	P	f	31	L	33	a	34
q	q	29	e	42	p	42	Q	Q	26	e	42	h	43
r	r	29	l	37	e	43	R	R	25	f	33	i	35
s	s	29	R	37	m	40	S	b	39	J	39	I	40
t	t	27	e	36	I	36	T	T	27	I	31	l	32
u	u	28	w	36	G	39	U	U	27	L	38	f	40
v	v	29	y	41	P	42	V	V	26	y	37	P	40
w	N	36	w	37	q	43	W	W	29	w	40	G	45
x	x	27	w	43	E	45	X	X	28	Z	39	v	40
y	y	26	P	38	K	40	Y	Y	27	e	38	M	41
z	z	29	A	35	H	36	Z	Z	27	I	34	e	38

 ISBN: 978-1-61208-154-0

Extracting Market Trends from the Cross Correlation between Stock Time Series

Mieko Tanaka-Yamawaki
Department of Information and Knowledge Engineering
Graduate School of Engineering, Tottori University,
Tottori, 680-8552 Japan, Japan
mieko@ike.tottori-u.ac.jp

Takemasa Kido
Department of Information and Knowledge Engineering
Graduate School of Engineering, Tottori University,
Tottori, 680-8552 Japan, Japan
s042026@ike.tottori-u.ac.jp

***Abstract*—We apply the RMT-PCA, recently developed PCA in order to grasp temporal trends in a stock market, on daily-close stock prices of American Stocks in NYSE for 16 years from 1994 to 2009 and show the effectiveness and consistency of this method by analyzing the whole data of 16 years at once, as well as analyzing the cut data in various lengths between 2-8 years. The extracted trends are consistent to the actual history of the markets. We also discuss on the problem of setting the effective border between the noise and signals considering the artificial correlation created in the process of taking log-return in analyzing the price time series.**

Keywords - RMT-PCA; Correlation; Eigenvalues; Principal Component; Stock Market; Trend.

I. INTRODUCTION

Recently, there have been wide interests on the use of RMT (Random Matrix Theory) in many fields of sciences [1-10]. In particular, the use of asymptotic formula of the eigenvalue spectrum of cross correlation matrix between independent time series of random numbers [11,12], as a reference to the corresponding spectrum derived from a set of different stock price times series in order to extract principal components effectively in a simple way [13-16], has attracted much attention in the community of econo-physics [17, 18]. The main advantage of this method as a principal component analysis is its simplicity. While the standard PCA (Principal Component Analysis) tells us to find the largest PC (Principal Component) and subtract this component from the entire data, and apply the same procedure recursively on the remaining data one by one, RMT-PCA (RMT-based principal component analysis) can process all the "non-random" components at once by subtracting the RMT formula from the eigenvalue spectrum of cross correlation matrix. Plerau, et al. [13] was one of the first attempts to apply this technique on stock price time series. By using the daily close stock prices of NYSE/S&P500, they successfully extracted eminent stocks out of massive data of price time series.

However, this method suffers from two difficulties. One is the restriction on the dimensionality, N, and the length of the data, T, such that $N < T$. Moreover, the entire set of N times T data are needed for analysis, since the basic quantity of analysis is the cross correlation matrix whose elements are the equal-time inner-products between a pair of stocks. Another difficulty is the restriction of the parameter size. Since the RMT formula is derived in the limit of N and T being infinity, we need a special care to keep the range of the parameters in which the RMT formula is valid.

By using machine-generated random numbers, such as rand(), etc., we have tested the validity of the RMT formula in various range of N and T, and have clarified that N=300, or larger, is the safe range unless T is not too close to N, and the validity decreases for smaller N, and the borderline is around $50<N<100$. Since the size of stocks dealt in the major markets exceeds 400, the applicability of RMT formula is justified.

Due to the restriction of the methodology to prepare the length of the time series, T, larger than the dimension of the correlation matrix, N, all the data extending to several years had to be combined into a single correlation matrix in [3-6], in which daily-close prices were used. Thus it was difficult to pin-point a short term trend or to compare trends of different time periods.

By employing intra-day (tick-wise) data containing all the transactions made every day, we can apply the methodology to the data of every year and compare the results of different years. We carried out the same line of study used in [13,14] by setting up the algorithm of RMT-PCA to be applied on intra-day equal-time price correlations. Based on this approach, we have shown that this handy methodology works well to extract the trend change of 4 year interval, from 1994 to 2002 [19,9].

In this paper, we apply the same algorithm to a wider set of stock price data including daily-close prices of American stocks in the database of S&P500 for 16 years from 1994 to 2009. We prepare the data of various lengths by cutting the 16 years into 2, 4 and 8 pieces and check the consistency and effectiveness of the proposed methodology.

II. EIGENVALUE PROBLEM OF CORRELATION MATRIX FOR STOCK PRICES

We shall briefly review the outline of the methodology used in RMT-PCA. The first step is to prepare the price time series into an $N\times(T+1)$ matrix named S, whose i-th row contains the price time series of length T+1. This matrix S is converted into a matrix of log-return as follows.

 ISBN: 978-1-61208-154-0

$$r(t) = \log(S(t+\Delta t)) - \log(S(t)) \quad (1)$$

We normalize each time series in order to have the zero mean the unit variances as follows.

$$x_i(t) = \frac{r_i(t) - < r_i >}{\sigma_i} \quad (i=1,\ldots,N) \quad (2)$$

The correlation $C_{i,j}$ between two stocks, i and j, can be written as the inner product of the two log-profit time series, $x_i(t)$ and $x_j(t)$,

$$C_{i,j} = \frac{1}{T} \sum_{t=1}^{T} x_i(t) x_j(t) \quad (3)$$

Here, the suffix i indicates the time series on the i-th member of the total N stocks.

The correlations defined in Eq. (3) makes a symmetric ($C_{i,j} = C_{j,i}$), square matrix whose diagonal elements are all equal to one ($C_{i,i} = 1$) and off-diagonal elements are in general smaller than one ($|C_{i,j}| \leq 1$). As is well known, a real symmetric matrix C can be diagonalized by a similarity transformation $V^{-1}CV$ by an orthogonal matrix V satisfying $V^t=V^{-1}$, each column of which consists of the eigenvectors of C. Such that

$$C v_k = \lambda_k v_k \quad (k=1,\ldots,N) \quad (4)$$

where the coefficient λ_k is the k-th eigen-value and v_k is the k-th eigenvector. This can be pursued by means of well-known Jacobi's rotation algorithm.

A criterion proposed in [3-6] and examined recently in many real stock data is to compare the result to the formula derived in the random matrix theory [1]. According to RMT, the eigenvalue distribution spectrum of matrix C made of random time series is given by the following formula [2],

$$P_{RMT}(\lambda) = \frac{Q}{2\pi} \frac{\sqrt{(\lambda_+ - \lambda)(\lambda - \lambda_-)}}{\lambda} \quad (5)$$

in the limit of $N \to \infty,\ T \to \infty, Q = T/N = \text{const}$ where T is the length of the time series and N is the total number of independent time series (i.e. the number of stocks considered). This means that the eigenvalues of correlation matrix C between N normalized time series of length T distribute in the following range.

$$\lambda_- < \lambda < \lambda_+ \quad (6)$$

Following the formula Eq. (5), between the upper bound and the lower bound given by the following formula.

$$\lambda_\pm = (1 \pm Q^{-1/2})^2 \quad (7)$$

The proposed criterion in our RMT_PCM is to use the components whose eigenvalues, or the variance, are larger than the upper bound λ_+ given by RMT.

$$\lambda_+ < \lambda \quad (8)$$

III. APPLICATION OF RMT-PCA ON STOCK PRICES

We prepare N normalized stock returns of the same length T, which makes a rectangular matrix of $S_{i,k}$ where i=1,…,N represents the stock symbol and k=1,…,T represents the traded time of the stocks. The i-th row of this price matrix corresponds to the price time series of the i-th stock symbol, and the k-th column corresponds to the prices of N stocks at the time k. We summarize the algorithm that we used for extracting significant principal components.

However, a detailed analysis of the eigenvector components tells us that the random components do not necessarily reside below the upper limit of RMT, λ_+, but percolate beyond the RMT due to extra randomness added in the process of computing the log-return in Eq. (1). Based on extensive numerical analysis, this percolation always occurs and the maximum front of the continuum spectrum extends to about 20% larger than the upper limit λ_+ of RMT. This fact suggests us that the upper limit λ_+ is not appropriate to separate the signal from the noise due to the percolation of the random spectrum over λ_+ but an effective upper bound λ_{eff} =1.2 λ_+ about 20% larger than the upper limit λ_+ of RMT. Then λ_+ in the step (4) of the RMT-PCA algorithm in Fig. 2 is to be replaced by λ_{eff}.

Algorithm of RMT-PCM :

(1) Select N stock symbols for which the traded price exist for all t=1,…,T, corresponding to all the working days of that term.
(2) Compute log-return r(t) for the selected N stocks. Normalize the time series to have mean=0, variance=0, for each stock symbol, i=1,…, N.
(3) Compute the cross correlation matrix C and obtain eigenvalues and eigenvectors.
(4) Select eigenvalues larger than λ_+, the upper limit of the RMT spectrum, and $\lambda_\pm = (1 \pm Q^{-1/2})^2$,
$P_{RMT}(\lambda) = \frac{Q}{2\pi\lambda}\sqrt{(\lambda_+ - \lambda)(\lambda_- - \lambda)}$
and identify those eigenstates as the principal components.
(5) Sort the eigenvector components corresponding to the eigenvalues identified in the step (4) above, in the descending order and identify the business sectors of the largest 20 components. If those 20 components belong to any particular sector, that is the leading sector in that term.

Figure 1. The algorithm to extract the significant principal components in RMT-PCA.

 ISBN: 978-1-61208-154-0

IV. TRENDS EXTRACTED AS THE EMINENT COMPONENTS OF EIGENVECTORS

We applied the algorithm stated in Section 3 on the daily-close prices of American stocks listed in S&P500, for 16 years from 1994 to 2009.

At first, the entire data of this period are used for analysis. Then the entire data is split to 2 parts, 1994-2001 and 2002-2009. Those are further split to 4 parts, 1994-1997, 1998-2001, 2003-2005, 2006-2009. Finally, they are split to 8 parts of 2years data, 1994-1995, 1996-1997, 1998-1999, 2000-2001, 2002-2003, 2004-2005, 2006-2007, 2008-2009. The results are listed in Table 1.

TABLE I. RESULTS FOR 16, 8, 4 YEAR DATA (EIGENVALUES LARGER THAN $2\lambda_+$ ARE HIGHLIGHTED IN BOLD=ITALIC)

	94-09	**94-01**	**02-09**	**94-97**	**98-01**	**02-05**	**06-09**
N	373	373	464	373	419	464	468
T	3961	2015	1946	1010	1002	1006	936
Q	10.6	5.40	4.19	2.71	2.17	2.17	2
λ_+	1.7	2.1	2.2	2.6	2.8	2.8	2.9
λ_1	***74***	***41***	***150***	***37.2***	***53***	***116***	***200***
λ_2	***11***	***13***	***15***	***8.7***	***19***	***14***	***18***
λ_3	***8.8***	***8.8***	***12***	***5.8***	***13***	***13***	***14***
λ_4	***7.7***	***6.9***	***11***	4.6	***9.2***	***9.1***	***8.9***
λ_5	***5.1***	***4.8***	***6.5***	3.3	***6.6***	***6.3***	***5.3***
λ_6	***4.3***	***4.2***	***5.1***	3.2	***5.8***	5.3	5.0
λ_7	3.3	3.5	3.8	2.8	4.7	4.8	4.4
λ_8	2.9	3.1	3.4	2.6	4.2	4.6	3.5
λ_9	2.5	2.7	3.3	2.4	3.8	4.0	3.2
λ_{10}	2.4	2.2	2.8	2.4	3	3.3	2.7
λ_{11}	2.0	2.2	2.4	2.3	2.8	2.9	2.7
λ_{12}	1.9	2.1	2.3	2.3	2.7	2.9	2.5

According to the step (4) in the RMT-PCA algorithm in Fig. 1 and , those 14 eigenstates are the principal components, based on . We find the business sectors of the companies of 20 largest components in the corresponding eigenvectors. If those components are concentrated in any particular business sector, we identify that sector as the trend makers during that time period. It can be proved mathematically that the eigenvector of the largest eigenvalue is consist of components of the same sign, and the corresponding sectors are not concentrated to a particular sector but distributed to any sectors, because the largest principal component show the global feature of the market thus corresponds to its representative index, such as S&P500, in our case of dealing with American stocks. The eigenvectors of the other eigenvalues have components of both signs. It has been known that the positive components and the negative components belong to the two separate business sectors, if they are strongly concentrated to particular sectors. Summing up those knowledge we have, the 2nd principal component reflects the trend of the time period of the data if any concentration of the sectors are observed.

The sectors are classified according to GICS (Global Industry Classification Standard) coding system, that classifies the business sectors of stocks into 10 categories. We denote them by a single capital letter, A-J as follows.

A: Energy, B: Materials, C: Industrials, D: Service, E: Consumer Products, F: Health Care, G: Financials, H: Information Technology, I: Telecommunication, and J: Utility.

If we take λ_{eff} instead of λ_+, as we explained in the last paragraph of Section 3, then we have 10 eigenstates corresponding to the eigenvalues λ_1=74.3,..., λ_{10} = 2.41, we have lesser number of principal components than the above-stated 14. However, the concentration of business sectors in the eigenvector components occurs only for the 4-5 largest eigenvalues and quickly becomes blur for smaller eigenvalues. Based on this observation, we might increase λ_{eff} to the range of $\lambda_{eff} = 2\lambda_+$, 100% larger than the theoretical criterion. In any case, the difference is irrelevant as long as we take only several principal components.We show 8 bars corresponding to

$$v_2(+),\ v_2(-),\ v_3(+),\ v_3(-),\ v_4(+),\ v_4(-),\ v_5(+),\ v_5(-),$$

where $v_k(+)/v_k(-)$ indicates the positive-sign part/negative-sign part of the vector of k-th principal component, by partitions corresponding to 10 sectors of A-J, and the corresponding eigenvalues and the sign of the components below each bar.

We observe from the graphs in Fig. 4 that the sector H (InfoTech) dominates the (+) components of v_2 and the sector J (Utility) dominates the (-) components of v_2.

The result of 8 years data, 1994-2001 and 2002-2009 are shown in Fig. 5, the left figure of which shows the dominance of J (Utility) and H (InfoTech) during the term 1994-2001, and the right figure shows the dominance of A (Energy) and G (Financials) during the term 2002-2009. This means the active sector has changed from J (Utility) and H (InfoTech) to A (Energy) and G (Financials) at the turn of the century.

The results of 4 year data, 1994-1997, 1998-2001, 2002-2005, and 2006-2009 are in Fig.6, showing the dominance of J (Utility) and H (InfoTech) both in 1994-1997 and 1998-2001, the dominance of A (Energy) and H (InfoTech) in 2002-2005, and A (Energy) and G (Financials) dominance in 2006-2009. The corresponding result of 2 year data is shown in Fig. 7. No clear structure is seen after 2002, except weak dominance of G (Financials) and A (Energy).

 ISBN: 978-1-61208-154-0

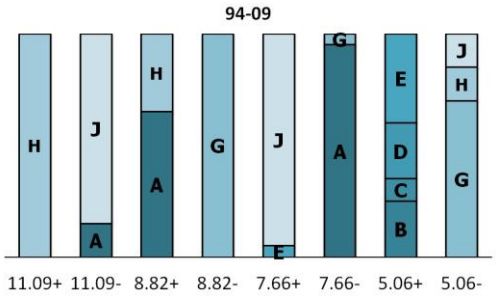


Figure 2. Trends of 16 years from 1994 to 2009 are shown. The sector H (Information Technology) and J (Utility) are the most eminent sectors in this period.

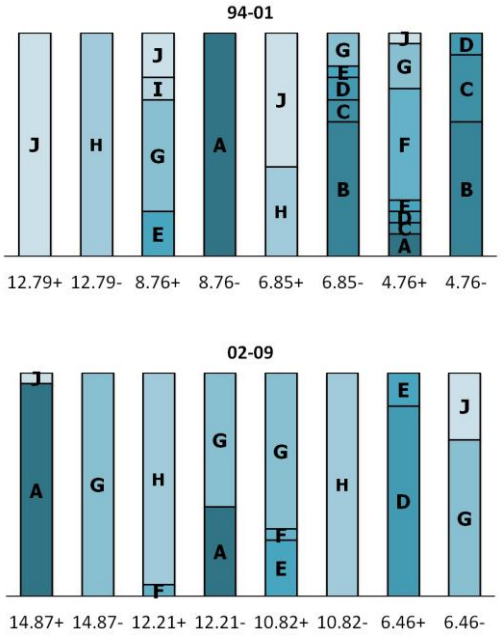
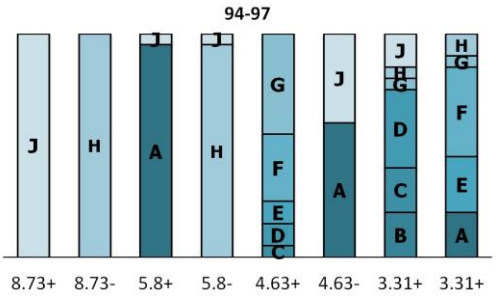
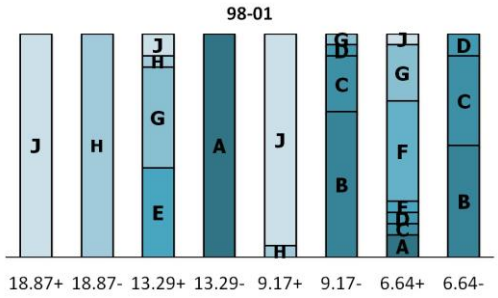


Figure 3. Trends of 8 years, 1994-2001 (left) and 2002-2009 (right). In 1994-2001, the sector J (Utility) and H (Information Technology) dominate, but in 2002-2009, A (Energy) and G (Financial) dominate the market.





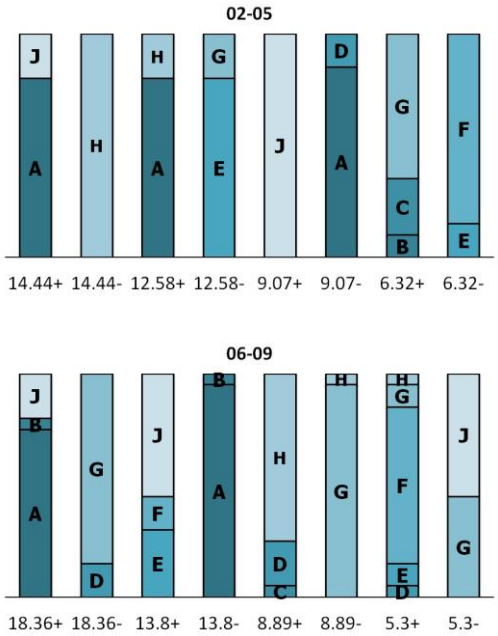
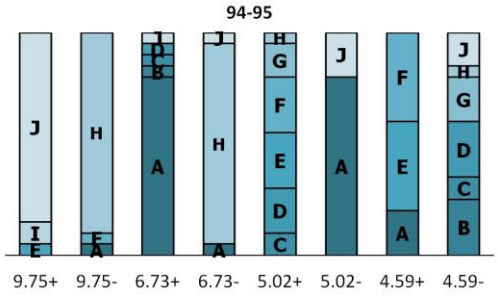


Figure 4. Trends of 4 years each are shown. Both in 1994-1997 and 1998-2001, J (Utility) and H (Information Technology) dominate, while A (Energy) and H (Information Technology)dominate in 2002-2005 and A (Energy) and G (Financial) dominate in 2006-2009.



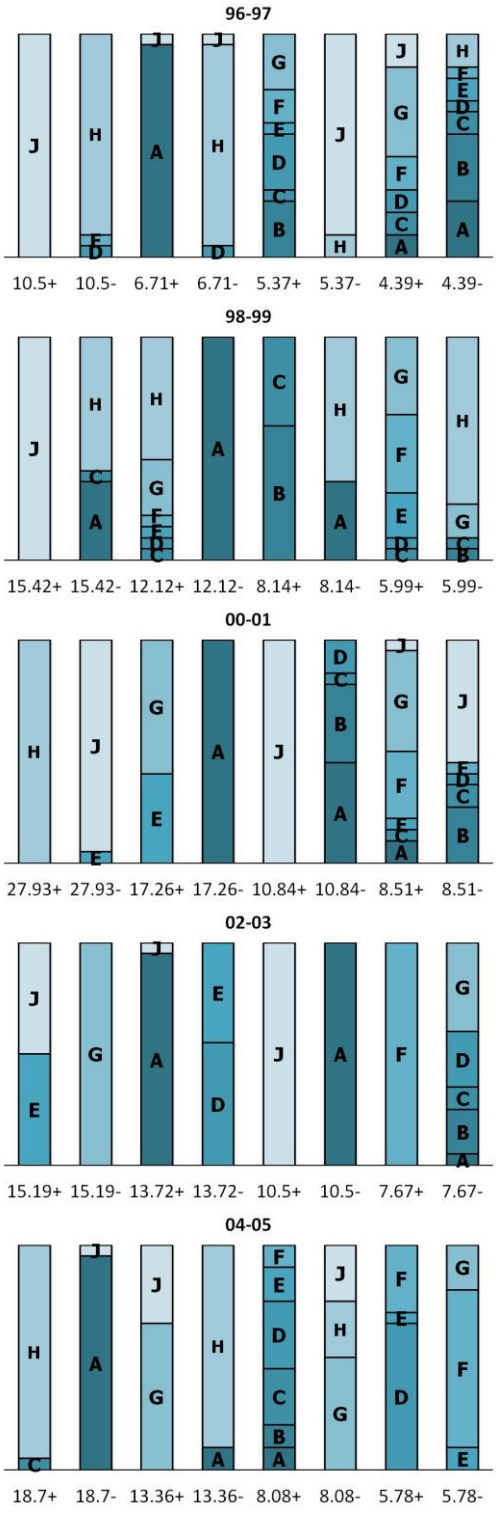


Figure 5. Trends of 4 years each are shown. Both in 1994-1997 and 1998-2001, J (Utility) and H (IT) dominate, while A (Energy) and H (IT) dominate in 2002-2005 and A (Energy) and G (Financial) dominate in 2006-2009.

V. CONCLUTION AND DISCUSSION

Our results have shown that the trend of each time period can be successfully depicted by the concentrated business sectors in the positive components and the negative components of the eigenvector corresponding to the 2nd principal components. Although the condition $\lambda > \lambda_+$ dramatically reduces the number of principal components compared to the conventional method of PCA. Moreover, our method is considerably simple with much shorter in process to extract principal components, which is a great advantage in the case of analyzing the stock market.

The conventional PCA tells us to extract the largest principal component and subtract this element from the entire data, and apply the same procedure recursively on the remaining data one by one. This kind of method requires a lot of computational time and is not suitable for analyzing a system of the large dimension, such as a set of stocks in the market. Another method of PCA uses the eigenvalues of the correlation matrix of times series, but tells us to pick up the components whose eigenvalues are larger than one, or the accumulated sum of eigenvalues exceeds 80 percent of the total sum, etc. Neither one is suitable for analyzing the stocks in the market, since the number of principal components thus obtained usually exceeds 100 for N=400-500, while the RMT- PCA has derived the number of principal components in the range of 5-13 in our lesson in Section 4 in this paper. We illustrate this point in Fig. 8. We also tabulate the numbers of principal components obtained in our analysis applied on 2 years data in Table 1.

 ISBN: 978-1-61208-154-0

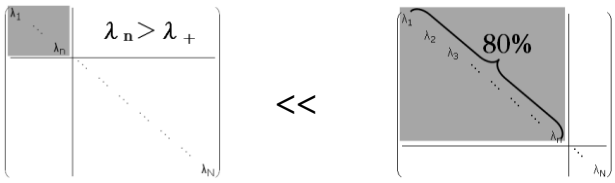


Figure 6. The RMT-PCA (left) offers much smaller number of PCs compared to the method of 80 percent accumulative eigenvalues (right).

Some remarks are in order before concluding this paper. Firstly, we mention the limitation of finantial data. The daily-close price data can be dowloaded on the web site, such as Yahoo Finance, etc. with free of charge, thus convenient for us to analyze. Moreover, the data are ready to use to calculate the equal-time correlation. Althogh some stocks are not traded every day, more than 400 stock symbols are traded every day in NYSE and TOPIX. However, the length of data per year is only 252, the working days of the markets. Thus we must combine two or more years to satisfy N < L.

On the other hand, the intra-day price data are sold commertially and quite costly. Moreover, the traded time for each stocks are not the same and we need pre-process the data in order to make the equal-time correlation. We have chosen the "block-tick" method to regard the trades within a certain block, such as every hour, to make them equal-time.

Finally, we mention the applicability of RMT-PCA on other fields of study. This methodology is not restricted for studying stock prices but can be applied for much wider variety of noisy data, including meteorological, or atomospheric data, and demographical data. Since the equal-time cross correlation matrix is the main tool of the methodology, we need to prepare a large number of equal-time data taken simultaneously at every moments, whose length L is larger than the number of the time series N, namely, N < L. Moreover, N > 300 is desirable by the reason explained in Section I.

Considering the time-delay for any information to propagates, however, the equal-time correlation is not sufficient, and we eventually need to consider the correlation with time delay. Technically speaking, however, the time-delayed correlation matrix C is not symmetric and the eigenvalues are not guaranteed to be real valued. The eigenvalue problem of such matrices is much more complicated compared to the equal-time correlation matrix, which is symmetric thus can be diagonalized by using the Jacobi's rotation algorithm.

References

[1]. Mehta, M. L., "Random matrices", 3rd edition, Academic Press (2004)

[2]. Edelman, Alan and Rao, N. Raj, "Random matrix theory", Acta Numerica, pp. 1-65 (2005)

[3]. Bai, Zhidong and Silverstein, Jack, "Spectral Analysis of Large Dimensional Random Matrices", Springer (2010)

[4]. Tao, Terence and Vu, Van, "Random matrices: universality of ESD and the circular law" (with appendix by M. Krishnapur), Annals of Probability, Vol. 38 (5), pp. 2023-2065 (2010)

[5]. Beenakker, C. W. J., "Random-matrix theory of quantum transport", Reviews of Modern Physics, Vol. 69, pp. 731–808 (1997)

[6]. Kendrick, David, "Stochastic control for economic models", McGraw-Hill (1981)

[7]. Bahcall, S. R., "Random matrix model for superconductors in a magnetic field", Physical Review Letters, Vol. 77, pp. 5276–5279 (1976)

[8]. Franchini, F., Kravtsov, V. E., "Horizon in random matrix theory, the Hawking radiation, and flow of cold atoms", Physical Review Letters, Vol.103, 166401 (2009).

[9]. Peyrache, A., Benchenane, K., Khamassi, M., Wiener, S. I., and Battaglia, F. P., "Principal component analysis of ensemble recordings reveals cell assemblies at high temporal resolution", Journal of Computational Neurosience, Vol. 29, pp. 309–325 (2009)

[10]. Sánchez, D., Büttiker, M., "Magnetic-field asymmetry of nonlinear mesoscopic transport". Physical Review Letters, Vol. 93, 106802 (2004).

[11]. Marcenko, V A, Pastur, L A, "Distribution of eigenvalues for some sets of random matrices", Mathematics of the USSR-Sbornik, Vol. 1, pp. 457–483 (1994)

[12]. Sengupta, A. M. and Mitra, P. P., "Distribution of singular values for some random matrices", Physical Review E, Vol. 60, pp. 3389-3392 (1999)

[13]. Plerou, V., Gopikrishnan, P., Rosenow, B., Amaral, L.A.N., and Stanley, H.E., "Random matrix approach to cross correlation in financial data", Physical Review E, Vol. 65, 066126 (2002)

[14]. Plerou, V., Gopikrishnan, P., Rosenow, B., Amaral, L.A.N., and Stanley, H. E., "Scaling behaviour in the growth of companies", Physical Review Letters, Vol. 83, pp. 1471-1474 (1999)

[15]. Laloux, L., Cizeaux, P., Bouchaud, J.-P., and Potters, M., "Noise dressing of financial correlation matrix", Physical Review Letters, Vol. 83, pp. 1467-1470 (1999)

[16]. Bouchaud, J.-P. and Potters, M., "Theory of Financial Risks", Cambridge University Press (2000)

[17]. Mantegna, R .N. and Stanley, H. E., "An Introduction to econophysics: correlations and complexity in finance", Cambridge University Press (2000)

[18]. Iyetomi, H. et al., Fluctuation-dissipation theory of input-output interindustrial relations, Physical Review E, Vol. 83, 016103 (2011)

[19]. Tanaka-Yamawaki, Mieko, "Extracting principal components from pseudo-random data by using random matrix theory", Lecture Notes in Artificial Intelligence, Vol. 6278, pp. 602- 611 (2010)

[20]. Tanaka-Yamawaki, Mieko, "Cross correlation of intra-day stock prices in comparison to random matrix theory", Intelligent Information Management (online journal: http://www.scrp.org) (2011)

 ISBN: 978-1-61208-154-0

Virtual Team Tasks Performance Evaluation Based on Multi-level Fuzzy Comprehensive Method

Dalia Kriksciuniene
Department of Informatics
Vilnius University
Kaunas, Lithuania
dalia.kriksciuniene@vukhf.lt

Sandra Strigunaite
Department of Informatics
Vilnius University
Kaunas, Lithuania
sandra.strigunaite@vukhf.lt

***Abstract* — The article analyses the research problem, what model for performance evaluation of virtual team could effectively serve a project leader by applying intelligent computational methods, which could reflect human cogitation. The hierarchical fuzzy comprehensive method of virtual team performance evaluation is suggested. The list of criteria of three groups (team, task and interaction) is elaborated for evaluation of project status and computed as the multi-level fuzzy vector. Expert evaluation, Shannon entropy and multi-level matrix computational methods are applied for experimental research of the model. The experimental results of comprehensive task evaluation by combining expert judgments and quantitative values derived from communication data are highly compatible and can assist project leader by providing valuable insights of underlying reasons affecting project outcome.**

Keywords - virtual team; task performance evaluation; human interaction management; multi-level fuzzy; comprehensive method.

I. INTRODUCTION

Globalization and penetration of information technologies are urging the need for new strategies of organizing projects in virtual mode and elaborating methods for their evaluation. Comparing to traditional attitudes the concept and principles of virtual work management are different not only due to the modes of communication in virtual and regular workspaces. Even experienced specialists and project leaders notify of numerous unexpected risks and difficulties, which undermine team efforts while working in virtual space. Recently, we can observe quite intensive informal discussions among project managers and leaders trying to reveal what brings the most significant influence in their efforts to streamline project progress, and what types of performance evaluation techniques could improve exposure of the real situation. This problem is important to project managers who apply both traditional and specific project methodologies. The group of team management problems in virtual environment is identified as e-leadership [1,2].

The traditional project management methodologies acknowledge administration and operational control. These techniques are characteristic for organizing work processes in the face-to-face environment, where prevailing methods of evaluation of progress status by the top-level organizational bodies are based on subjective insights. High level of subjective judgments and uncertainty makes project evaluation costly, time consuming and not all the time effective.

The process-based method Project Management Body of Knowledge (PMBOK) [3] accentuates the interaction-oriented tools and techniques for project team management: *Observation and Conversation, Project Performance Appraisals, Conflict Management, and Issue Log*.

Emerging project management approaches (such as Agile methodologies) emphasize importance of leadership in management process. Leader is assumed as the person who keeps the spotlight on the project vision, who inspires the team, who promotes teamwork and collaboration, who champions the project and removes obstacles hindering progress [4]. One of the most highly evaluated qualities of team leader is the ability to identify teamwork situation and project progress status. The effective tools and models for evaluation of the teamwork characteristics could enhance impact of personal qualities and to apply proper leadership techniques.

Human interaction management theory (HIM) presents different perspective of human work modeling principles and suggests specific notation for revealing interaction processes and their content among team members [5].

The interaction analysis oriented methods emphasize communication processes among team member, but do not provide evaluation tools, except plain statistics of communication intensity.

The issues of performance evaluation of the virtual teamwork are summarized in [6], where the possibility to derive interaction statistics-based variables and to find their causal relationships for evaluating project performance is explored by applying balanced scorecard approach. The intelligent analysis of virtual communication variables by combining neural networks and sensitivity analysis in [7] reveals that the information captured in project environment cannot be directly applied for evaluation of project members, but the combined neural network-based analysis of the derived variables can predict project outcomes with sufficient precision, while the impact of various variables is different. The feasibility of introduction of interaction–based variables and fuzzy rules of their interrelationships for comprehensive evaluation of project team performance is researched in [8].

 ISBN: 978-1-61208-154-0

The article analyses the research problem, what model for analysis of virtual team could serve as effective evaluation tool for project leader, and to provide insights of performance by applying intelligent methods, which could reflect human cogitation.

In Section 2, we present analysis of pilot survey, which explores attitudes of experienced project managers towards performance evaluation. The survey outcomes enabled to suggest new method of creating hierarchical structure of criteria for performance evaluation of virtual project. As the experts denoted it is crucial to evaluate virtual team performance based on members' interaction analysis with the help of a system of interrelated criteria. In Section 3, we explain the structure and computational procedures of the model, designed by applying multi-level fuzzy comprehensive method, and aimed to identify and calibrate weights of task evaluation criteria. The conclusion and future works section summarizes the outcomes of the research and evaluation of adequateness of the model to the findings of pilot survey.

II. IDENTIFICATION OF PERFORMANCE EVALUATION CRITERIA

The research of criteria for evaluating performance of virtual team was made in two phases: the survey of project experts and computational analysis by applying method of multilevel fuzzy assessment.

In order to find out different attitudes and reveal significant insights to project performance evaluation fifteen project managers from different information technologies-related companies, participated in the interviewing process. The participants of the discussion were project managers of highest experience and technical consultants of JIRA Agile and JIRA Confluent solutions. They had experience of using ComindWork [9] and other applications of similar functionality for supervising virtual teams. The virtual collaboration environment ComindWork allows tracing various types of interactions among the members of project teams: messages, submitted files, timing settings and assignment of task status during project cycles [9].

The interviews consisted of two parts. The questions of the first part were formulated according to the main principles of Human Interaction management theory: Team building, Communication, Knowledge, Empowered time management, Collaborative, and Real time planning.

The second part of survey aimed to collect detailed information that could indicate difficulties of virtual project management and lack of e-leadership efficiency related to performance evaluation and situation assessment. One of the goals of the discussion was to find out which of the criteria could be derived and measured by using logging records of virtual collaboration interactions, such as ComindWork and further processed by the computational model.

After summarizing interviews the list of criteria for virtual team task performance evaluation based on human virtual interaction, was generated. After the reflective analytical discussion, the initial list of evaluation criteria was shortened. The refined version of the criteria list for human interaction evaluation is presented in Table 1. The criteria were grouped into three parts: *Team* evaluation criteria, *Task* evaluation criteria and *Interaction* evaluation criteria (Table 1). The *Team* and *Interaction* evaluation criteria were designed to evaluate performance either of teams or of individuals.

TABLE I. VIRTUAL TEAMWORK EVALUATION CRITERIA

Criteria and Description
Team evaluation criteria
Size- number of performers assigned to task during whole task implementation period (describes Team)
Variety - level of different roles assigned to task (describes Team)
Experience - level of team experience (describes Team and Individuals)
Characteristics - cumulative measures of personal characteristics: attitude to work/task implementation of the performer (describes Team and Individuals)
Hierarchy - level of team hierarchy (the rate of high, middle, junior experience within a particular role); (describes Team)
Task evaluation criteria
Phase - expert judgment /manager evaluation, parameter rate from a particular interval (beginning, middle or end of a particular project phase or iteration)
Task intelligence level - expert evaluation of human driven effort necessary to implement the task
Difficulty - expert evaluation of the level of task difficulty
Result clarity - expert / project team evaluation about the expected results and quality criteria
Organizational level - managers' evaluation of the strength of organizational structure of the team
Interaction evaluation criteria
Meeting level - expert evaluation (qualitative component) Information captured of system event logs (quantitative component, defining duration, number of topics, etc.) (describes Team)
Questioning level - number of questions sent to team members and requests for information to team leader and senior members (describes Team and Individuals)
Information sharing level - eagerness of team and individual members to share information (describes Team and Individuals)
Activity level - activeness level of the individual and team (describes Team and Individuals)
Punctuality level - punctuality level of the individual and team(describes Team and Individuals)
Maturity level - expert evaluation of the maturity level, which encompasses all characteristics (describes Team)

The second part of the research was executed together with project managers, virtual project environment application specialists and associated statistics research professionals. The goal of the second part was to develop or to adapt classical methods for creating computational measurement system for human interaction criteria. Everybody agreed that the most suitable criteria for *Team* evaluation were traditional articles of project management evaluation that have been deeply investigated by researchers and successfully adopted in practice by project managers. For deriving these criteria the liner mathematical calculations were suggested and applied.

The group of *Task evaluation* criteria was strongly based on expert knowledge and different task implementation circumstances that could not be accepted and analyzed as standard, as they were never known in advance. Therefore, all criteria in *Task* group were evaluated by expert judgment.

 ISBN: 978-1-61208-154-0

The measurement of third group *Interaction evaluation* criteria was based on application of computational methods for processing data acquired by event recording in collaboration system by Interaction observing agents. For example, analysis of the parameter *Meeting level* accumulated information about the number of meetings, which were held during implementation of particular task, the duration of meetings, their types according to goal, planning time, number of participants and results. Parameters *Questioning level* and *Information sharing* level depend on number of messages and chats distribution according to their purpose types, also meetings held on a particular question (problem) or learning issues. *Activity* level summarized total amount of actions done by individuals and teams during task implementation period. *Punctuality level* referred to fact if interim or final results, reviews of tasks were fulfilled in time, whether the response to question was done with acceptable time delay, if there were no late arrangement of meetings. The last parameter *Maturity level* served as compound parameter, which accumulated all values from *Interaction* evaluation criteria group and two criteria of *Team* evaluation group (*Experience* and *Hierarchy*). The whole parameter set is presented in Table 2.

TABLE II. TEAM EVALUATION PARAMETERS

Input Criteria	*Team / Individual*	*Type*	*Measurement / Rate interval*
Team evaluation criteria			
Size (Rsz)	Team (T)	Quantitative	(Rsz) = ΣN(i) N-participant
Variety (Rvr)	Team (T)	Quantitative	(Rvr) = (ΣR(i)/N+ ΣR(n)/N)
Experience (Rex)	Individual (I) / Team (T)	Qualitative	(Rex) = (ΣR(i)L(i)/N+ ΣR(n)L(n)/N) R-role type L-experience level
Characteristics (Rch)	Individual (I)/Team (T)	Qualitative	Rate from interval [1…100]
Hierarchy (Rhr)	Team (T)		Rate from interval [1…10] (Rhr) = Σ(R(i)/L(i))
Task evaluation criteria			
Phase (Tph)	-	Qualitative	Rate from interval [1…n] accordingly to the project planning strategy
Task intelligent level (Tint)	-	Qualitative	Rate from an interval [-100…100];
Difficulty (Tdf)	-	Qualitative	Rate from an interval [0…100];
Result clarity (Trc)	-	Qualitative	Rate from an interval [0…50];
Organizational level (Iorg)	-	Qualitative	Rate from an interval [1…5];
Interaction evaluation criteria			
Meeting level (Ime)	Team	Quantitative / Compound	Rate from an interval [1...n];
Questioning level (Oque)	Individual / Team	Quantitative/ Compound	n- depends on task time duration and project team members;
Information sharing (Iish)	Individual / Team	Quantitative/ Compound	
Activity level (Iact)	Individual / Team	Quantitative/ Compound	
Punctuality level (Ipun)	Individual / Team	Quantitative/ Compound	
Maturity level (Ima)	Team	Statistical/ Compound	Rate from an interval [1…5]; (Ima) = f(Ime, Oque, Iish, Iact, Ipun)

The project management experts agreed that skillful observing team communication in virtual settings were no less significant and challenging as in the real environment.

The main information used by project leader to derive insights were team behavior patterns, interpretation of situations and task implementation processes by reviewing and evaluating content and intensity of team interactions. For example, analysis of team members' qualification and activities, questioning level, tendency to obey or to avoid obligations, punctuality in doing everything on time/not on time), keenness to share information and other characteristics could make it possible to identify the potential risk of task failure or possible delay. By employing insights from compound evaluation characteristics, the team leaders used to reduce risk of false judgment and make corrective actions of the situation concerning team behavior.

The presented model was further investigated by applying experimental data in order to compare the evaluation of team performance by skillful experts and by computational intelligence methods for defining values of the selected criteria.

Four possible situation outcomes of teamwork performance were chosen and experimentally tested by applying evaluation procedures of the suggested model:

(1) If task implementation is proceeding well;

(2) If the risk of task implementation delay can be identified;

(3) If the problem of non-understanding and chaos is accruing;

(4) If the project work is stagnated or omitted.

The computational method based on multilevel fuzzy approach for criteria estimation, the experimental setting, procedures of application the suggested model and evaluation of its results are discussed in the Section 3.

III. TASK PERFORMANCE EVALUATION MODEL BASED ON MULTI-LEVEL FUZZY METHOD

The method presents the new conceptual model of ability to integrate numerous criteria of different hierarchical levels for deriving status of the highest-level criteria. Multi-level Fuzzy Comprehensive method was chosen because of the variety of task evaluation criteria and their causal relationships. The quantitative evaluation by extracting information of virtual project environment is possible only

 ISBN: 978-1-61208-154-0

for deriving part of the low-level criteria. The expert evaluation of the low-level criteria can be done with sufficient quality even by project leaders with lower experience, whereas expert judgment of high level criteria implies high risk and subjectivity. The method is based on Fuzzy analytic hierarchy process approach [10, 11], which allows formation of hierarchical criteria matrix, defining values of the low-level criteria and deriving values of the highest level criteria by calculating their weights and probabilities of status according to the defined status vector.

The experimental results are based on analysis of the performance of software solution implementation project team, which consisted of seven members. The database of interactions among team members in the virtual environment ComindWork consisted of 937 records, which were analyzed for evaluation of team performance.

The results of the expert evaluation for each level of criteria were benchmarked to the results obtained by applying the suggested model.

Task Performance evaluation of the Multi-level Fuzzy Comprehensive methods were applied by these steps:

(1) The determination of all level criteria sets;
(2) The identification of each criteria evaluation;
(3) The indication of weight for each criterion;
(4) The construction of single criteria evaluation matrix;
(5) The construction of comprehensive evaluation model.

A. *The determination of all level criteria sets*

Virtual team evaluation criteria are shown in Table 2. Variables differ by the following characteristics: category (applied for team, task or interaction), application type (applied for team or individual), measurement principles (qualitative, quantitative or compound) and measurement methods (Table 2). The set of criteria is arranged as the matrix C=(Cij), (i=1,2, … m), (j=1,2,…n), where (i) indicates higher level category and (j) category of lower hierarchical level.

B. *The identification of each criteria evaluation*

Evaluation of each criteria group is done according to the expert recommendations and specific results. The criteria evaluation set is constructed by applying status rating vector V = (V1, V2, V3, V4) = {Omit; Chaos; Delay; Well}.

C. *Weight indication for the criteria*

The weight values for each criterion were determined by two methods: they were defined by experts and by applying Shannon entropy measure [9].

The weight indicator corresponds to each criteria level W=(Wij), (i=1,2,..m), and (j=1,2,..n). The list of criteria and their weights are shown in Table 3, where (i) indicates first level criteria and (j) – the second level criteria. Weight parameters should meet several conditions: Wi, Wij >0, Wij < 1, and formula (1):

$$\sum_{i=1}^{m} Wi = \sum_{j=1}^{n} Wij = 1 \qquad (1)$$

In order to research if the quantitative measures of interaction can be computed and used to assist the project leader (and at least partially replace expert judgment), the Shannon entropy measure was applied [12]. Shannon entropy takes into account the information effectiveness of the analyzed data series and can be applied for defining weights of the criteria. We applied standard algorithm for calculating weights of interrelated criteria [10]. Our suggested list of criteria (Table 2) consisted both of expert judgment-based values and of quantitative measures, defined from human interaction statistics in virtual collaborative space.

TABLE III. LIST OF CRITERIA AND THEIR WEIGHTS

Category	*Weight*	*Criteria*	*Weight*
Team characteristics (C1)	0.2	(1) Team Size (C11)	0.15
		(2) Role variety (C12)	0.25
		(3) Experience (C13)	0.3
		(4) Characteristics (C14)	0.2
		(5) Team hierarchy level (C15)	0.1
Task characteristics (C2)	0.3	(6) Phase (C21)	0.1
		(7) Task intelligent level (C22)	0.2
		(8) Difficulty (C23)	0.1
		(9) Result clarity (C24)	0.2
		(10) Organizational level (C25)	0.4
Interaction characteristics (C3)	0.5	(11) Meeting level (C31)	0.1
		(12) Questioning level (C32)	0.15
		(13) Information sharing (C33)	0.15
		(14) Activity level (C34)	0.1
		(15) Punctuality level (C35)	0.2
		(16) Maturity level (C36)	0.3

We applied Shannon entropy for computing weights for the interaction-statistics-based criteria for the individual members and the project team (C13, C14, C32, C33, C34, and C35 from Table 3). We implied that the interaction data provided by the virtual collaborative space of the team can be applied for calculating part of the evaluation criteria set and provide insights for the project leader for further evaluation of the whole criteria set.

The results of weight assignment according to Shannon entropy measure, as presented in Table 4, are highly compatible to the weights set by experts (Table 3).

TABLE IV. WEIGHTS COMPUTED BY SHANNON ENTROPY METHOD

Criterion (C)		*Weight (W)*
C13	Rex	W1/0,273
C14	Rch	W2/0,121
C32	Oque	W3/0,237
C33	Iish	W4/0,106
C34	Iact	W5/0,130
C35	Ipun	W6/0,133

The differences in evaluation may have occurred because the experts put their weight values based on their long-term

 ISBN: 978-1-61208-154-0

experience, and the Shannon's entropy method was applied to process the interaction data of a single project. The results encourage for further investigations aimed achieving validity of the experimental research.

Application of Multi-level Fuzzy Comprehensive method was intended to analyze experimental data in order to exploit hierarchical interrelationships of the criteria set and expert evaluations of the particular criteria in order to compute cumulative evaluation the team performance.

D. The construction of single criteria evaluation matrix

The single criteria evaluation matrix is constructed by using Matrix scheme for two or more indicators (2):

$$Rij = \begin{bmatrix} R_{i11} & R_{i12} & \dots & R_{i2v} \\ R_{i21} & R_{i22} & \dots & R_{i2v} \\ \dots & \dots & \dots & \dots \\ R_{in1} & R_{in2} & \dots & R_{inv} \end{bmatrix} \quad (2)$$

The evaluation matrix for the criteria set presented in Table 3 consists of expert grades. The following matrices R_{1j}, R_{2j}, R_{3j}, express expert judgment of probabilities of each situation outcome (3):

$$R_{1j} = \begin{bmatrix} 0{,}1 & 0{,}4 & 0{,}2 & 0{,}2 \\ 0{,}1 & 0{,}4 & 0{,}3 & 0{,}1 \\ 0{,}4 & 0{,}1 & 0{,}1 & 0{,}4 \\ 0{,}3 & 0{,}4 & 0{,}1 & 0{,}2 \\ 0{,}2 & 0{,}3 & 0{,}3 & 0{,}2 \end{bmatrix} \quad R_{2j} = \begin{bmatrix} 0{,}1 & 0{,}3 & 0{,}3 & 0{,}3 \\ 0{,}3 & 0{,}1 & 0{,}3 & 0{,}3 \\ 0{,}1 & 0{,}4 & 0{,}4 & 0{,}1 \\ 0{,}1 & 0{,}4 & 0{,}3 & 0{,}2 \\ 0{,}5 & 0{,}2 & 0{,}1 & 0{,}2 \end{bmatrix}$$

$$R_{3j} = \begin{bmatrix} 0{,}2 & 0{,}3 & 0{,}2 & 0{,}2 \\ 0{,}4 & 0{,}3 & 0{,}2 & 0{,}1 \\ 0{,}2 & 0{,}2 & 0{,}3 & 0{,}3 \\ 0{,}2 & 0{,}3 & 0{,}3 & 0{,}2 \\ 0{,}2 & 0{,}1 & 0{,}4 & 0{,}3 \\ 0{,}1 & 0{,}3 & 0{,}4 & 0{,}2 \end{bmatrix} \quad (3)$$

E. The construction of Comprehensive evaluation

Comprehensive evaluation is multi-level process. The computation is performed starting with the lowest level criteria of the initial criteria set Ci = (Ci1, Ci2 … Cij), then processing to the highest level criteria C = (C1, C2, Cm). The comprehensive evaluation matrix for Ci is Bi, for C is B, as computed by (4) and (5):

$$B_i = W_i * R_v = (b_{i1}, b_{i2}, b_{i3}) \quad (4)$$

and

$$B = W * R = W * \begin{bmatrix} B_1 \\ B_2 \\ B_3 \end{bmatrix} = (b_1, b_2, b_3) \quad (5)$$

The value vector V was applied for expert judgments. The computation results of the three criteria groups C1, C2 and C3 are evaluated according to values of vector V: (Omit; Chaos; Delay; Well) = {V1, V2, V3, V4}. The biggest element of the computed array means that the value of the computed criterion is equal to the corresponding element of the vector V with the highest probability. After applying the above-discussed computation procedures, the evaluation parameters for *Team*, *Task* and *Interaction* level characteristics (C1, C2, and C3) were computed and the project status evaluation was derived.

- *Team* characteristics:
  B1 = W1 * R1 = (0,24;0,30;0,19;0,24)
- *Task* characteristics:
  B2 = W2 * R2 = (0,3;0,25;0,23;0,22)
- *Interaction* characteristics:
  B3 = W3 * R3 = (0,2;0,25;0,34;0,22)
- *Comprehensive project task* evaluation:
  B = W * R = (0,24; 0,26; 0,28; 0,22).

The results lead to further interpretations and insights. The *Team* characteristic of the second level of Comprehensive evaluation was assigned value "Chaos" with the highest possibility (0,3). The *Task* characteristic was evaluated as "Omit" with the highest possibility (0,3). Analysis of *Interaction* denotes the situation of "Delay" (0,34). By accumulating all parameters according to multi-level fuzzy comprehensive evaluation method and calculating the compound situation, measure B the situation "Delay" could be identified with the highest probability (0,28).

Slightly different situation is identified in the case of applying calculation of criteria weights by Shannon entropy method, where the values of criteria subset are derived from interaction statistics information captured in the virtual project environment (Table 4). In this case the compound situation measure "Delay" has higher possibility equal to 0,29, as the B {*Omit; Chaos; Delay; Well*} = (0,24; 0,27; 0,29; 0,20).

The research results revealed that the suggested multi-level interaction system and expert judgment lead to similar results and can be applied to assist project leaders working in virtual environment. The human interaction instances registered during virtual teamwork can be used for deriving evaluation criteria and provide reliable insights for evaluating team performance and its outcomes. The initial results encourage that further research should be executed.

IV. CONCLUSION AND FUTURE WORK

The main drawback of applying existing methods (Agile, PMBOK, HIM) for project management in virtual environment is lack of measures for performance evaluation.

The analysis of virtual communication among project team members is mainly based on statistics of communication intensity, captured in the virtual project environment. The information of number of messages, submitted files and responses is not sufficient for providing in-depth insights for the project leader for comprehensive

 ISBN: 978-1-61208-154-0

evaluation of task quality, individual input or risk of project outcomes.

The article presents solution for the research problem as the multi-level fuzzy comprehensive evaluation model for performance evaluation of virtual team. The variables and their multi-level structure were elaborated by applying method of expert survey. The performance of the model is based on measurement of the variables and assigning their weights indicating their impact for the project outcome. Sixteen criteria of three groups (team, task and interaction) were included to the model for evaluation of project status, computed as the multi-level fuzzy vector. The measurement of criteria is combined not only of expert evaluation but of quantitative values related to communication of project team members as well. Shannon entropy method is applied for processing communication-related information captured in virtual environment of project team.

The experimental research of consisting of pilot expert survey and analysis of project teamwork performance in virtual environment ComindWork allowed to positively evaluated the suggested model. The values of interaction statistics-based criteria processed by applying Shannon entropy were highly compatible to the values assigned by experts.

It allows concluding that at least part of criteria for project performance evaluation can be computed by processing information of virtual communication environment used by the project team. Project leaders can evaluate the other part of low-level qualitative criteria included to the hierarchical model. The highest level-criteria for evaluating situation and forecasting the project outcome involve the biggest risk and require mature experience of project leader.

The suggested method of multi-level fuzzy assessment allows applying computational model, to calculate the highest level-criteria and assist project leader by providing valuable insights of underlying reasons affecting these criteria.

The proposed multi-level fuzzy method is an innovative approach for project performance evaluation by creating hierarchical interrelationship structure of criteria and their measurement by combining expert evaluation and Shannon entropy. The suggested method based on fuzzy logic can provide better understanding to project outcomes considering ambiguous and imprecise data in a manner similar to the human thinking and the human judgment.

The validity of the model will be tested by future experimental research.

REFERENCES

[1] D. N. Den Hartog, A. E. Keegan, and R. M. Verburgb, “Limits to leadership in virtual contexts,” The Electronic Journal for Virtual Organizations and Networks, vol. 9, Special Issue “The Limits of Virtual Work”, July 2007.

[2] M. R. Lee, “E-leadership for project managers: A study of situational leadership and virtual project success,” Ph.D. Dissertation, Capella University, USA, 2010, ISBN 9781124067209

[3] A guide to the project management body of knowledge: PMBOK guide. 3rd ed. Pennsylvania: Four Campus Boulevard. ISBN 1-930699-45-X.

[4] Agile project management IV. The Means: An Agile Project Management Framework Agile Project Management, Pace Systems, 2011.

[5] K. Harrison-Broninski, "Human Interactions: The Heart and Soul of Business Process Management," Meghan-Kiffer Press 2005, ISBN 0929652444.

[6] D. Kriksciuniene and I. Sarkiunaite “Influence of virtual environment on a human factor within the virtual teamwork,” Transformations in business and economics, 2007, vol. 6, no. 2, suppl. A, pp. 219-234, ISSN 1648-4460.

[7] D. Kriksciuniene and V. Sakalauskas, “The individual performance measurement framework in virtual team learning,” CSEDU 2010: proceedings of the 2nd international conference on computer supported education, Valencia, Spain, April 7-10,2010, vol. 2, pp. 150-155, ISBN 9789896740245

[8] S. Strigunaite and D. Kriksciuniene “Self-adapting intelligent business processes execution analysis,“ Business information systems workshop: BIS 2010 international workshops, Berlin, Germany, May 2010: revised papers, Book Series: Lecture Notes in Business Information Processing, 2010, vol. 57, pp. 29-32.

[9] ComindWork, Manage people online, http://www.comindwork.com, [last access 22/07/2011]

[10] Q. Zhao and W. Song. “DOM Quality Evaluation Based On Multi-Level Fuzzy Comprehensive Assessment Of Entropy Weights,“ The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, vol. XXXVII. Part B4. Beijing, pp.1457-1460, 2008.

[11] T. L. Saaty, “The Analytic Hierarchy Process,” McGraw-Hill International, New York, 1980.

[12] C. E Shannon, “A Mathematical Theory of Communication,” Bell System Technical Journal, vol. 27, pp. 379–423, 623-656, July, October, 1948

 ISBN: 978-1-61208-154-0

The Development and Implementation of a Short Term Prediction Tool Using Artificial Neural Networks

Aubai Alkhatib
University of Kassel
REMENA
Kassel, Germany
alkhatibaubai@yahoo.com

Siegfried Heier
University of Kassel
REMENA
Kassel, Germany
heier@uni-kassel.de

Melih Kurt
Fraunhofer IWES
Kassel, Germany
mkurt@iset.uni-kassel.de

***Abstract*—Wind speed forecasting is an essential prerequisite for the planning, operation, and maintenance works associated with wind energy engineering. This paper attempts to forecast fluctuations based only on observed wind data using the data-driven artificial neural network approach. Wind fluctuations with varying lead times ranging from a half year to a full year are predicted at Al-Hijana, Syria with the pre-preparation for the available data. Two layers of feed-forward back-propagation networks were used along with the conjugate gradient algorithm and other tested training functions. The results show that artificial neural network models perform extremely well as low values of errors resulting between the measured and predicted data are obtained.** The present work contributes to previous work in the field of wind energy independent power producer market and may be of significant value to Syria, considering that the country is currently in the process of transitioning into a free energy market. It is likely that this modeling approach will become a useful tool to enable power producer companies to better forecast or supplement wind speed data.

Keywords-Artificial Neural Networks; Wind Speed; Mean root square error; Forecasting.

I. INTRODUCTION

The wind-energy spread usage in recent years is an attempt to address the environmental problems that result from the consumption of energy and especially from nuclear power plant disasters like the one that recently occurred in Fukushima, Japan. The IPCC (Intergovernmental Panel on Climate Change) indicated that [1] human activities are directly related to increased atmospheric levels of greenhouse gasses, i.e., carbon dioxide, methane, chlorofluocarbons, and carbon monoxide. Additionally, a correlation also exists between global warming involving greenhouse gas and environmental problems. It is generally agreed that of those harmful greenhouse gasses, carbon dioxide contributes the most to global warming. The main artificial source of carbon dioxide discharge is derived from fossil fuels (conventional power plants). Therefore, much recent research has focused on reducing the consumption of fossil fuels and replacing those with renewable, environment-friendly energy sources. Currently, wind energy is considered as one of the most promising energy sources. However, since wind is difficult to manage, generating wind energy is still a challenge. Due to a variety of factors, the wind speed characteristic curve can change with time. Wind blows as a result of an imbalance in the quantity of heat on the earth by the energy from the sun. Experimentally, it is known that wind speed is intermittent, irregular, and frequently fluctuates in the short term. Since wind energy is directly related to the cubic value of the wind speed, any changes in the wind speed will greatly impact the amount of the energy. In order to better support the transition to a free energy market, a more accurate means of estimating the energy generated from the wind farm and pumped in the grid is needed.

This paper thus introduces an ANN (Artificial Neural Networks) for wind speed predictions to estimate the wind speed in a suggested location in Syria that involves two main approaches.

1- A one year prediction tool.
2- A half year prediction tool.

Also, the different possible ways that can be used in order to improve the prediction output (e.g., choosing different training functions). This paper also introduces a model for energy estimation using the output of the wind speed prediction tool as an input for the energy model. Using the Matlab computing program for building the suggested ANN is one of the future computing methods for wind prediction.

II. WIND SPEED PREDICTION TECHNIQUES

A. Introduction

The wind speed characteristic can be considered a non-linear fluctuation. Therefore, the forecasting of this function using traditional methods can be very difficult and time consuming. In this case, the intelligent engineering represented by a neural network, a chaos fractal, and a genetic algorithm, etc. can be applied. While these techniques are already adopted in numerical predictions, the usage of the ANN gives a better performance in terms of pattern recognition and finding location peculiarities, especially when information on the used wind turbine and power curve is given [2].

There are two different types of wind speed predictions [3]:

The vertical wind speed prediction or the prediction of the expected wind speed curve in one point on the geographical map with different height. This can be seen, for example,

 ISBN: 978-1-61208-154-0

when the wind measurement device is at a height of 40 m and the wind turbine is installed in the same location yet in a different hub height like 105 m.

The horizontal wind speed prediction or the prediction of the expected wind speed curve in one point on the geographical map that has a horizontal difference from the point of measured data. This is witnessed when the wind measurement device is in one location and the wind turbine is installed in another location (top of a mountain where the measurement is very difficult to be obtained) [4].

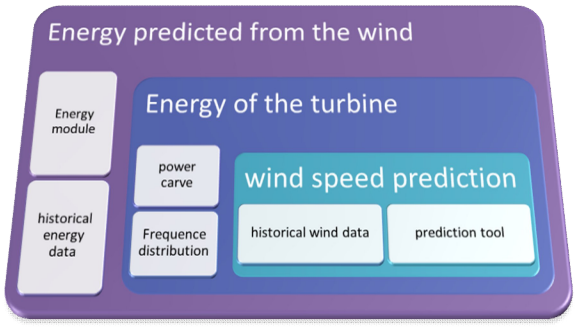


Fig. 1 Wind speed prediction and energy module connections

The historical wind data shown in Fig. 1 indicates that the atmospheric parameter measurements such as the pressure and temperature that were available for the location in our case. The energy historical data indicates a previous energy output for a previous wind turbine installed in the location of interest which was not available in our case (as this was the first wind farm to be installed in this location).

B. *Feed Forward Neural Network with Backpropogation*

A neural network is a computational structure that resembles a biological neuron. It can be defined as a "*massively parallel distributed processor made up of storing processing units, which has a natural propensity for storing experimental knowledge and making it available for use*"[5].

A feed-forward neural network consists of layers. Every layer will be connected to the previous once with more than one connection that has a weight to determine the importance of this connection. Every network has at least three layers. These include the input layer, output layer, and the hidden layer(s). The strength of a set of inputs can be determined by the activation function after adding the whole input signals as shown in Fig. 2.

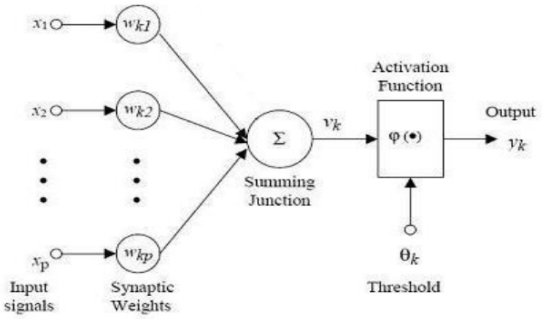


Fig. 2 Basic structure of a neuron [5]

The raw data was provided in a form of Excel file. Patterns were generated and a statistical analysis performed to get a good correlation among the input values. Some data was fed as an input in the prediction network for training purposes while other data was specifically employed for network testing purposes.

The following steps were taken to get the wind speed prediction:

1- Data Acquisition & Pre-processing.
2- Data conversion & Normalization.
3- Statistical Analysis.
4- Design of the Neural Network &Training.
5- Testing.

C. *Data Acquisition, Pre-processing, and Data Conversion*

The weather parameter values were collected at a weather station at the location of interest [6]. Time series was provided for every ten minutes with the help of the Syrian National Energy Center for Research and Development. The values of three different parameters were utilized to include the pressure, temperature and wind direction as shown in table I.

TABLE I LIST OF NETWORK PARAMETERS[8]

station	day	hour	speed 40 s wvt	direct 40 d1 wvt	direct 40 sd1 wvt	speed 40 max	speed 40 std	speed 10 avg	speed 10 max	speed 10 std	temp	pressure
14	1	0	1.656	199.8	8.55	2.01	0.183	1.184	1.57	0.142	3.344	947.24
14	1	10	1.549	185.6	9.74	1.91	0.164	1.136	1.47	0.097	3.581	947.1
…	…	…	…	…	…	…	…	…	…	…	…	…
…	…	…	…	…	…	…	…	…	…	…	…	…
14	366	2230	2.803	278.1	8.74	4.71	0.643	2.579	3.73	0.48	3.624	945.58
14	366	2240	3.069	282.2	7.22	4.29	0.385	2.593	3.55	0.379	3.392	945.68

During the data acquisition stage, the maximum value among each parameter was computed and normalization was carried out for all of those parameters [7].

 ISBN: 978-1-61208-154-0

D. Statistical Analysis

Since the amount of available data is massive and the characteristic curve of the wind speed continually changes with time, a statistical analysis is needed in order to measure the extent of the relationship between each of the meteorological values and to get rid of the redundant values that might be present in the data set. Therefore, a "*Spearman rank correlation*" was applied. The amount of correlation in a sample (of data) is measured by the sample coefficient of correlation, generally denoted by 'r' or by 'ρ,

E. Spearman's Correlation

Spearman's correlation allows testing the direction and strength of a relationship [9]. For example the relationship between the pressure and the wind speed will be shown (one of the inputs of the prediction tool and the output) to help determine the importance of this parameter on the output of the prediction. This in turn can give a good vision of the expected output of the suggested ANN tool. This approach can also be applied to problems in which data cannot be measured quantitatively but in which a qualitative assessment is possible. In this case, the best individual is given rank number 1, the next rank 2, etc.

The correlation coefficient takes values between [1,-1]. A value of /1/ indicates that the relationship between the two different parameters is very strong and has a positive effect (when "X" increases, the "Y" value will also increase). The value/-1/ has the same strength meaning of /1/ yet the relation is inverse. A value of /0/ means that no relationship exists between the two different studied parameters.

Steps for achieving a Spearman's ranking:

A- Rank both sets of data from highest to lowest value (make sure to check for tied ranks /readings of the same value and to obtain the same sequence of readings).

B- Subtract the two sets of ranking data to get the difference /d/.

C- Square the values of /d/.

D- Add up the squared values of the differences.

E- Calculate the values using Spearman's Ranking Formula [9]:

$$R = 1 - \frac{6 \times \Sigma D^2}{n(n^2-1)} \qquad (1)$$

Table II shows the results obtained from this analysis for one year data (2008). It can be seen that the pressure has an inverse influence on the wind speed and that the temperature has an indirect effect on the wind speed through an inverse relationship with the pressure [12].

TABLE II SPEARMEN'S RANKING RESULTS FOR 2008

2008				
Correlation	Wind Speed	Direction	Temperature	pressure
Wind Speed	1			
Direction	0.232188687	1		
Temperature	0.257124942	0.214808385	1	
pressure	-0.49489753	-0.29900903	-0.70568132	1

Fig. 3 gives a statistical analysis for 6 years of available data. The figure can be used to clarify the results obtained from the prediction tool as it shows that the atmospheric parameters and the character curve of the wind speed changes on a yearly basis. The information obtained from Spearmen's ranking can help determine which data should be selected as training data in order to contain the best possible situation and get better prediction results for this specific location.

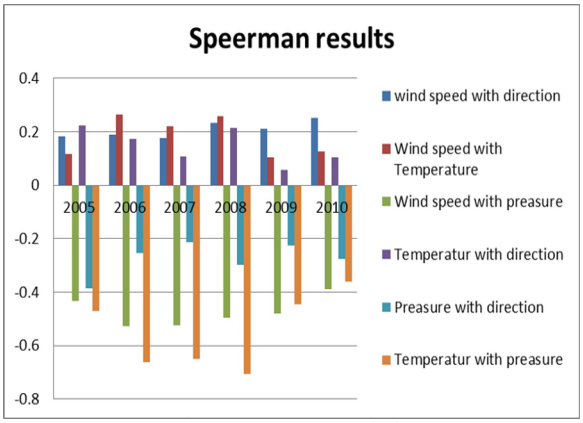


Fig. 3 Spearman's analysis

F. Design of the Neural Network & Training

Designing the neural network means sizing the network in order to fit our need. Unfortunately, there is currently no mathematical equation for sizing the network or determining which training functions to use [10]. Thus, engineers often rely on trial and error and personal experience to solve these issues. In our case, the sizing of the number of hidden layers, training functions, activation functions, number of neurons in the hidden layers, and determining the best training input pattern was accomplished through trial and error and, as shown in Fig. 4, a comparison of the results. Finally, 2 hidden layers with feed forward activity were chosen. Using the back propagation algorithm in each training set, the weights were modified in order to reduce the root mean squared error (deviation) (RMSE/D/) between the predicted values and the actual readings as target values. Thus, the modification takes place in the reverse direction from the output layer until the terminating condition is reached. The steps are:

- Initialize the weights.
- Propagate the inputs forward.
- Back propagate the error.

 ISBN: 978-1-61208-154-0

- Terminating condition.

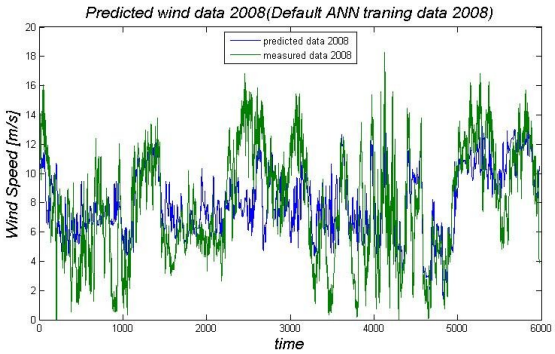


Fig. 4 Deviation of predicted and measured wind speed for 2008

G. Testing

Testing is the final stage needed to finalize the proposed wind speed prediction tool. While different methods can be used to evaluate the results obtained from the prediction tool, in this case the Mean Square Error method was used [11].

TABLE III RMSE OF THE WIND PREDICTION TOOL WITH DIFFERENT INPUT POSSIBILITIES FOR DIFFERING YEARS.

different inputs with time																				
Description	max					min					avarage					RMSD				
Years	2004	2005	2006	2007	2008	2004	2005	2006	2007	2008	2004	2005	2006	2007	2008	2004	2005	2006	2007	2008
Wind data input	0	17.33	17.49	7.866	11.82	0	-7.81	-9.74	-10.2	-7.73	0	-0.08	1.141	-0.016	1.4833	0	0.062	0.0824	0.051	0.07493
all as input	0	11.57	15.19	5.447	11.49	0	-16.8	-16.3	-13.7	-9.81	0	-5.6	-2.21	-3.79	-0.0325	0	0.139	0.1214	0.092	0.05967

TABLE IV RMSE OF THE WIND PREDICTION TOOL WITH DIFFERENT TRAINING DATA FOR DIFFERING YEARS.

two year input for prediction with time tool																				
Description	max					min					avarage					RMSD				
Year	2004	2005	2006	2007	2008	2004	2005	2006	2007	2008	2004	2005	2006	2007	2008	2004	2005	2006	2007	2008
two year 2007-2008	NP	15.68	15.21	8.52	9.099	NP	-17.5	-13.9	-7.88	-10.4	NP	-0.36	-1.6	0.098	0.065	NP	0.098	0.057	0.038	0.057
two year 2006-2007	NP	17.61	15.89	8.731	15.56	NP	-22.7	-8.23	-8.03	-10.6	NP	-0.42	-0	0.257	5.677	NP	0.159	0.0441	0.068	0.127

TABLE V RMSE OF THE DIFFERENT YEARS WIND PREDICTION TOOL WITH DIFFERENT TRAINING FUNCTIONS.

different training functions																				
Description	max					min					avarage					RMSD				
Years	2004	2005	2006	2007	2008	2004	2005	2006	2007	2008	2004	2005	2006	2007	2008	2004	2005	2006	2007	2008
Bayesian Regulation	0	16.86	14.09	12.35	10.1	0	-10.1	-11.1	-13	-10.7	0	-0.66	-0.45	-0.712	-9E-05	0	0.075	0.0791	0.075	0.06691
Fletcher-Reev	0	17.2	13.02	8.371	11.14	0	-8.95	-10.4	-10.2	-8.42	0	-0	-0.39	-0.703	0.0462	0	0.074	0.0786	0.071	0.066973
Marquardt	0	20.42	12.62	11.05	9.874	0	-18.2	-9.73	-10.1	-8.66	0	1.594	-0.5	-0.913	-0.063	0	0.155	0.0787	0.068	0.066888
Quasi-netwton	0	16.92	13.24	11.34	10.49	0	-9.04	-9.38	-9.98	-8.51	0	-1.24	-0.49	-0.864	-0.055	0	0.074	0.0786	0.069	0.066996

The previous tables give the results of the differing sizing possibilities that can be used for the prediction tool. It can be seen that the usage of the pressure, temperature and wind direction as inputs for the prediction tool is more effective than using each parameter alone. Also, the 2007-2008 input data gives better results than the 2006-2007 data because the MSE is better in the first case.

The most important conclusion that can be derived from the data in the previous tables is that if the error is reduced a new approach is needed in order to overcome the errors generated from the unaccounted character changes of the wind speed. In this new approach is to take only half a year into account. Also this half year was divided among collecting testing (or validation) data and training data. The half year period was divided into days, with one day allocated for training and the next one for testing and so on as shown in Fig. 5.

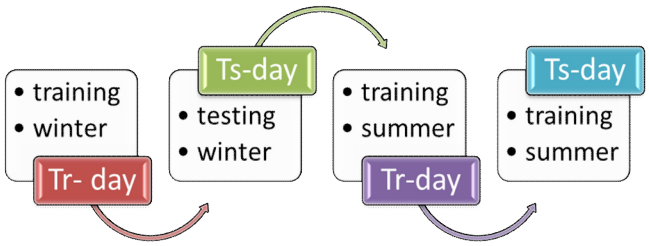


Fig. 5 The new wind prediction tool (Tr= training, Ts= testing).

Fig. 6 compares the results of the old approach with the results of the new one. The first two columns shows the results of the old approach along with the best results from the different input data and training data respectively, and the second two columns show the same results but for the new approach. An error of /RMSD=0.0449/ was obtained.

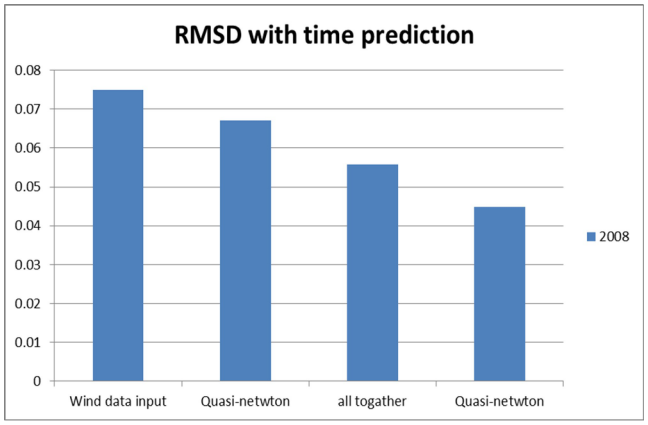


Fig. 6 The error reduction due to the usage of the new approach.

Although the results shown in Fig. 6 only describe one year (2008), if applied to more than one year as shown in Fig. 7 it can be seen that the RMSD for the old approach is better than the new approach. However, since the purpose of this research is for energy calculations and the energy market (in other words, for engineering not meteorological applications) the approach needs to have a very small error. For this reason, the new approach can be considered more effective in this situation as shown in Fig. 6 or Fig. 8, which show the deviation between the measured and predicted wind speed.

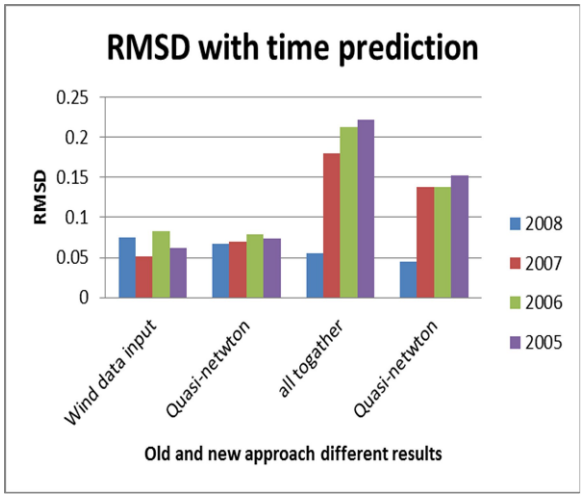


Fig. 7 Comparison of the RMSE for the old and new approach for differing years.

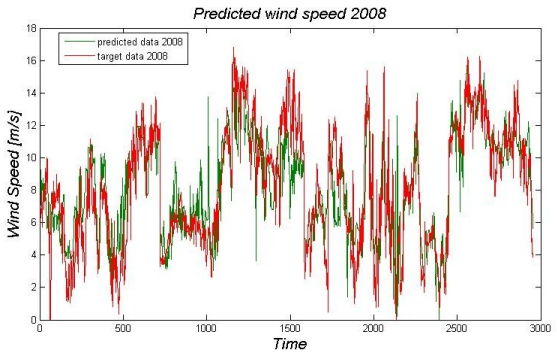


Fig. 8 Measured and predicted wind speed for 2008 using the new approach.

H. Energy Module

After getting acceptable results as an output from the previously built prediction tool, those results are used as an input for the energy module. Fig. 9 shows the suggested energy model which has the following components:

1. A signal Builder: contains the predicted wind speed data.
2. Lookup Table: contains the power curve data of the used power turbine [12].
3. Integrator: is used to get the output energy from the wind turbine [13].
4. Scope: is used to show the results in Fig. 11.
5. Display: is used to show the accumulated value of the energy.

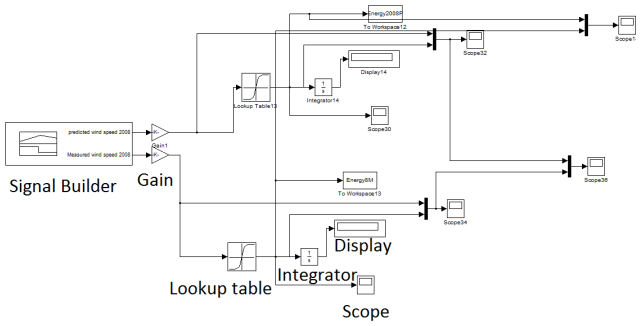


Fig. 9 Block diagram of the energy module.

Putting the previous component together give us the energy that can be produced by the used turbine.

Vestas Wind System /V90/ was selected as the working wind turbine with a 90 m rotor diameter, 105 m height, and 2 MW power [8]. Fig. 10 shows the power curve of this wind turbine.

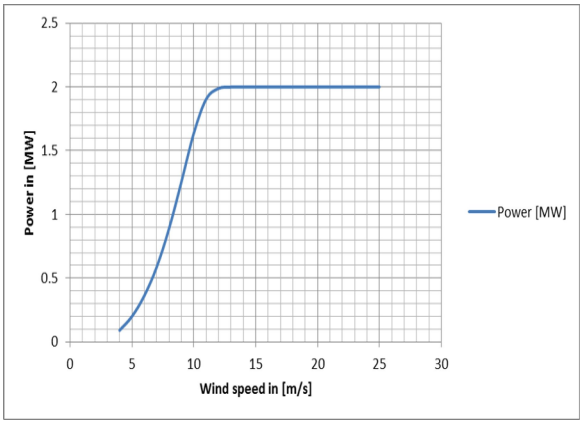


Fig. 10 Power curve of V90 wind turbine.

Using all of above information, the energy can be obtained as shown in Fig. 11 where the error is tripled due to the relation between the energy and the cubic wind speed. For this reason, the error of the predicted wind speed should be at its minimum with no time deviation errors [14][15].

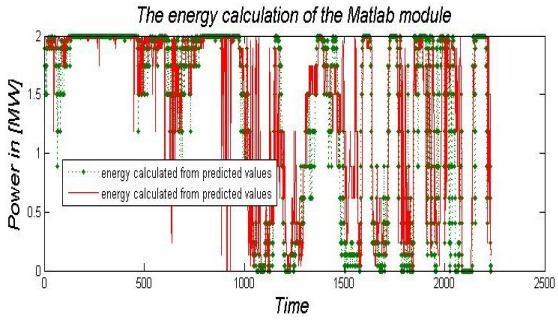


Fig. 11 The difference between the energy calculated from the predicted and measured wind speed

III. CONCLUSION:

As a conclusion of this work it is observed that the prediction errors can be reduced by the usage of same characteristic properties of the predicted wind speed and that in its turns will lead to less errors in the Energy module (the error of the energy is cubical to the wind speed errors due to the cubic relationship between the Energy and the wind speed). More understanding of the data lead to better results in the prediction tool that is why spearman's analysis is an important method of determining the strength of connection between the different data used as input to get the output of the prediction tool. Finally, the short term prediction helps of reducing the errors in the prediction tool and also the work load on the computing device.

REFERENCES

[1] J.T. Houghton, L.G. Meira Filho, B.A. Callander, N. Harris, A. Kattenberg and K. Maskell " Climate Change 1995, The Science of Climate Change, Contribution of WGI to the Second Assessment Report of the Intergovernmental Panel on Climate Change" Published for the Intergovernmental Panel on Climate Change, Cambridge University Press, 1995, pp. 27-73.

[2] Dr. Kurt Rohrig, Rene Jursa "Online-Monitoring and prediction of wind power in German transmission system operation centers" Königstor 59, D-34119, Kassel: IWES, pp. 1, 2002.

[3] Dr. Matthias Lange, Dr. Ultich Focken" State of the art in wind power prediction in Germany and international developments"Marie-Curie-str.1, D-26129 Oldernburg: Oldenburg uni.,pp. 2-3, 2009.

[4] Tony Burton, David Sharpe, Nick Jenkins, and Ervin Bossanyi "Wind energy handbook" London: John Wiley & Sons, Ltd., 2001.

[5] Manoj Kumar" SHORT-TERM LOAD FORECASTING USING ANN TECHNIQUE" Rourkela-769008: National Institute of Technology,pp. 9, 2009.

[6] Ministry of Electricity, "Request for qualification (RFQ)" Damascus: Syrian Arab Republic, 2009.

[7] Sathyajith Mathew" Wind Energy Fundamentals, Resource Analysis and Economics" Springer, 2007.

[8] Decon "Pre-feasibility study AL hijana. Damascus" Syrian Energy Center, Damascus, Syria, 2005.

[9] S.Chand. Managerial statistcs. AMIT ARORA, 2009.

[10] Juan R.Rabunal and JulianDorado "Artificial neural network in real-life application" London: Idea Group Publishing, 2006.

[11] Wikipedia®. (2010). http://en.wikipedia.org/wiki/Root_mean_square_deviation. Retrieved 11,21, 2010, from www.wikipedia.com: www.wikipedia.com

[12] K.Sreelakshmi, P. Ramakanthkumar" Neural Networks for short term wind speed prediction" World Academy of Science ,Engineering and technology, pp.724, 42 2008.

[13] Alok Kumar Mishra and L. Ramesh "Application of Neural networks in wind power (Generation) Prediction" IEEE, pp. 3, 2008.

[14] Mituharu Hayashi and Bahman Kermanshahi "Application Artificial neural network for wind speed and determination of wind power generation output" Nakamachi, Koganei-shi,Tokyo 184-8588, Japan: Tokyo Uni.,pp.2-3, 2009.

[15] M.C. Mabel and E.Fernandez" Estimation of Energy Yield from Wind Farms Using Artificial Neural Networks" IEEE, pp. 3, 2009.

 ISBN: 978-1-61208-154-0

Process Management Reviewed

Mohsen Sharifi[1], Seyedeh Leili Mirtaheri[1], Ehsan Mousavi Khaneghah[1]
[1]School of Computer Engineering
[1] Tehran, Iran
{msharifi, mirtaheri, emousavi}@iust.ac.ir

Zeinolabedin Mosavi Khaneghah [2]
[2]Faculty of Management, University of Tehran
[2] Tehran, Iran
zmousavi@pmamut.com

***Abstract*—We propose a new 5-layered pyramid of process needs that must be administered by local and global managers of the process society, namely the computer operating systems of the future. We also propose a new needs-oriented approach to process management based on the proposed categorized process needs. We argue that in contrast to traditional blind order-oriented operating system process managers, needs-oriented process-aware operating system process managers are more favorable to future computing environments with relatively large-scale orders of magnitudes of processes and resources scattered in smallest and biggest imaginable scales and varieties.**

Keywords-process; process management; process needs; operating systems; process-aware

I. INTRODUCTION

Computing systems have long entered and instrumented human societies in recent decade so much so that most are advocating electronic societies in which computers act on behalf of humans. These actions are thus quite vital to human well being and require meticulous criticism.

We believe in the revival of human understanding of and attitude towards computer entities that actually denote these actions namely *processes*. We advocate a U-turn on our viewpoint on event-driven computer processes, replacing the old black-box view by a more autonomous process-aware view wherein each process has a repetitive though limited life cycle of providing services to categorized discrete requests from outsiders.

We envisage a more reasonable management of locally and globally distributed computer processes than traditionally achievable. In the same way Abraham Maslow [1, 2] has categorized human needs in a pyramidal hierarchy, we categorize and structure process needs in 5 layers. In this hierarchy, self-consciousness is at the top and vital needs at the bottom of the pyramid, and security and social and power needs are in between, for the purpose of better management of societies and their needs [3].

Processes need recognition by the process society to begin with. This implies that each process must be granted a globally unique identifier as a first citizen entity and enough local and global space to start living. Having got the vital resources from local and global administrators and managers of the process society, they require proper means for inter and intra process communications to autonomously and in their own discretion pursue their goals and objectives prescribed at their birth time.

There are no aimless processes in the process society and processes need to communicate and cooperate to achieve their goals in their limited lifetime. In the race for resources, processes may wish to get more privileges from the process society compared to other processes in order to get to their goals. The managers of the society must thus provide some sort of improvising the priorities.

Processes get more self conscious and aware of their own behaviors and their society as time passes and they come closer to their termination time. Although not all processes might be concerned with their security, they are all vigilant on their safety all along their lifetime.

We thus envisage a 5-layered pyramid of process needs that need to be administered by local and global managers of the process society, namely the computer operating systems of the future. We can now draft a new philosophy for the management of processes based on the given categorized needs in terms of lifetime-needs. We argue this philosophy is more favorable than traditional process management of operating systems to future computing environments with relatively unbounded large-scale orders of magnitudes of processes and resources that are scattered in smallest and biggest imaginable scales.

The rest of paper is organized as follows. Section II presents the traditional definition of processes and how they are used to be managed in operating systems. Section III argues in favor of a change in the traditional view and proposes a new set of definitions and roles for processes and process managers. Section IV argues how the propositions can benefit future computing and Section V concludes the paper.

II. TRADITION

Believing in a purely and fully mechanical world, we humans are used to envisage a manufacturing factory as a complex of (electro) mechanical tools operated by a number

 ISBN: 978-1-61208-154-0

of human (and more recently, humanoid) operators to produce as many products as possible at the lowest costs. Executive managers on the factory floor take their orders from top management and monitor and control these orders are exactly followed by the operators to the last point with no regard at all to the consciousness and awareness of operators on what and how they really do the work. The prominent paradigm is purely order-oriented. Operators and executive managers that take orders best irrespective of how they feel about it or whether orders satisfy their requirements are considered as best. This is what used to be followed and advertized by the adverts of Taylor school [4, 7].

The operators and executive floor managers are the active entities, i.e., *processes* and *process managers* respectively, in this game. Those who have originally designed the factory, i.e., *creators*, though humans but are not active in the runtime manufacturing process and are not willing to sacrifice production rate by any unanticipated runtime requirements of operators or executive floor managers. Even worth, they do not grant much authority to executive floor managers to manage operator society differently than already prescribed at design time.

This Taylor style of management only works fine in cases where everything is known correctly and completely (i.e., accurately and exactly) a priori and proper design is sought based on this valid and complete knowledge, does not requiring to bother about any unforeseeable changes at all. This condition had never been satisfied in real world though especially in a working set including humans as their principal active entities of work.

We have experienced a similar dogmatism in the management style of computer operating systems too. Problem domain experts eager to use computer were forced to think they know the exact solution to their exactly known problems. Programmers were forced to think the advocated solutions by expert domains were exact, which had to be followed in every prescribed step and sought states. Operating systems followed suit and supposed that programmers exactly knew the constraints of operating systems and the hardware they ran on. They thus executed *processes* running these programmed solutions in the exact orderly-prescribed manner. This worked fine as long as no change was made to any suppositions. The executive floor manager (i.e., *process manger* of operating system) controlled the exact orderly-prescribed execution of processes without any regards for any possible process needs. This is to say that it could have denied the basic needs of a process but still expect the orderly execution of the process.

We do not live in utopia anymore [2, 9]. We are currently living in an information era with very unknowns but yet many hard requirements to be able to work and service irrespective of our many unknowns. Nowadays, domain experts no more claim to have the exact knowledge about their problems and solutions to these problems a priori. Programmers are no longer rest assures that they have programmed the exact solutions to problems. Processes expect to face unforeseen cases not orderly-programmed before. Process managers have to manage ever-increasing resource hungry processes that race for additional new types of needs too (e.g., membership, safety, security). Process communities are getting worldwide and wider under disparate communities. New varieties of resources scattered and distributed in different administrative domains have emerged. Voluminous data is generated as time passes which needs to be fed to processes at runtime with many runtime constraints. At last but not the least, requirements are changing fast in all dimensions and at all levels. In short, we live in a dynamic compute world.

This new compute world requires a completely different approach and style to management of its constituent active entities, i.e., *creators*, *process managers*, and *processes*. It is no more possible for designers to prescribe orderly-prescribed execution of processes and expect process managers to handle all changes in process needs by themselves.

In the next section, we present a purely managerial solution to the general problem and then present our proposition accordingly for operating system *process managers*, *processes*, and *creators.*

III. PROPOSITIONS

Maslow has presented a new more conscious management style that is more akin to work in an ever-changing real world. He has rightly replaced the old purely *order-oriented* approach with a *needs-oriented* approach to management, categorized in a five level hierarchical pyramid (Fig. 1) [5, 6]:

1. ***Physiological Needs*:** for food, drink, air, sleep-the basic bodily "tissue" requirements.
2. ***Safety Needs*:** for security, stability, protection from harm or injury; need for structure, orderliness, law, predictability; freedom from fear and chaos.
3. ***Belongingness and Love Needs*:** for abiding devotion and warm affection with spouse, children, parents, and close friends; need to feel a part of social groups; need for acceptance and approval.
4. ***Esteem Needs*:** for self-esteem based on achievement, mastery, competence, confidence, freedom, independence; desire for esteem of others (reputation, prestige, recognition, status).
5. ***Self-Actualization Needs*:** for self-fulfillment, actually to become what one potentially can be; desire to actualize ones capabilities; being true to ones essential nature; be what one can be.

In short, the Maslow school of management believes that humans (a la *processes*) can be expected to act more responsibly and effectively towards community goals only if their evolutionary declared needs are recognized in their societies (*creators*, *process managers*, and *other processes*) and are incrementally satisfied by their communities in their lifetime.

 ISBN: 978-1-61208-154-0

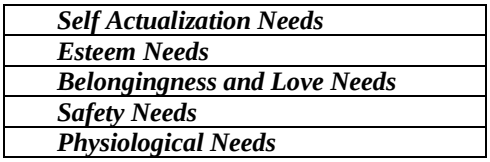


Figure 1. Maslow human needs hierarchy [3, 8].

Taking Maslow's managerial approach as a base, we have envisaged and propose a 5-layered hierarchical pyramid called Melz-Mazlow pyramid for compute process needs in the compute world of today too (Fig. 2), whereby processes are considered as the smallest active entities executing solutions to domain experts' problems (operating system processes are excluded):

1. ***Existential Needs*:** for storage space (memory, cache, disk, and file), and processing time (CPU, GPU, and Core) – the "vital" basic requirements.
2. ***Safety Needs*:** for process security, stability, protection from harm, attacks, and injuries; need for structure, orderliness, law, predictability; freedom from fear and chaos.
3. ***Openness and Belongingness Needs*:** for process communication mechanisms (IPC, DSM, MP) to converse with child processes, parent processes, process managers, and other processes outside local domains; need to become a member of and communicate with other compute processes and social groups; need for acceptance and approval.
4. ***Competitiveness Needs*:** for process self-esteem and survival based on race conditions, achievements, mastery, competence, confidence, freedom, independence; desire for esteem and referential position amongst other processes (priority, leadership, reputation, prestige, recognition, status) and desire for the good of other processes (binding, cooperation, trust, friendship, richness, upgrade, in addition to their reputation, prestige, recognition, and status) too;
5. ***Transcendence Needs***: for process self-awareness, self-fulfillment, actually to become what a process can potentially be and is expected to achieve at its limits; desire of process to actualize fully its capacities and capabilities; being true to ones essential nature; be what a process can be – the ultimate "*behavioral* self-consciousness" requirements.

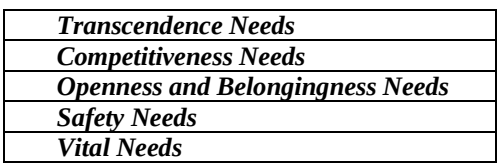


Figure 2. Proposed Melz-Mazlow compute process needs hierarchy.

Using the proposed Melz-Mazlow 5-layered process needs, we now propose a request-based approach in place of an order-based approach to process management in operating systems of the future.

To begin with, it is essential to reiterate and remind ourselves about some of the important best practices in operating systems design in the past that lay the best grounds for design and implementation of any future operating system, including its process management:

1. Operating systems need not get involved in policies but rather should take the policies as input and try to enforce them as best as required by making the most intelligent use of compute resources; process managers must follow suit and avoid from getting involved in deciding on policies on behalf of processes.
2. Operating system kernels must be kept as primitive as possible just to satisfy the bare requirements not all requirements.
3. Operating system designers and developers are not engaged during execution of processes explicitly. However, they could implicitly influence the executions of processes by embedding runtime mechanisms to enforce their intensions. In other words, *creators* of operating systems are not engaged in day-to-day operations of processes. Operating systems act on behalf of their creators.
4. Process managers of operating systems are synonymous to factory floor executive managers that manage operating processes running solutions to domain expert problems.
5. Operating systems can perform more effectively to satisfy the requirements of processes if they are made aware of the overall requirements of processes well in advance of executions of processes either statically or dynamically but with the least overhead on process management and total performance of the system.

The question is now how does a process manager that is intended to work based on process needs differ from traditional process managers that take orders from their creators blindly with no regard for process needs. The answer is simple: process manager must be reflective to process needs.

Information on each process are stored in kernel structures as before but just an snapshot of these information is given to the process upon every request of process so that it can look at these information and get informed about its status quo so that to guide the process manger what to do to best satisfy its needs. This way the process knows by itself and takes the responsibility that it is not reasonable to ask for its safety needs if it had not asked for its vital needs yet or its previous calls for vial resources had not been satisfied yet. In turn, it does not engage itself in communication with other processes if it has not asked for protection yet or that it had asked for it before but not provided yet by the process manager or other managers concerned. Only an open process enabled to communicate asks for socialization with other

 ISBN: 978-1-61208-154-0

processes to strengthen its competitive edge in its own local society or in other global societies. Finally, only a process that is conscious about its position in communities will be willing to cater and ask for its transcendence needs.

Therefore, what we propose here is that processes behave more wisely and orderly by themselves and allow the process manager to take their more thoughtful needs and manage them all more wisely as guided by processes. This is quite in contrast to traditional processes that go wild to ask for as much as possible of everything irrespective of their status quo, burdening the process manager with how to resolve many possibly conflicting requirements and to satisfy the blindly requested needs of all processes.

The gaining of the process manager is tremendous at the cost of merely providing the current image of information it has on a process to the process upon handling of every request of the process. The trick is that process manager becomes process aware whilst it is still kept primitive with the least thickening of the operating system kernel.

Interestingly, such a reflective process manager is expected to be scalable to handle global distribution and higher varieties and numbers of process needs.

IV. ARGUMENT

Let us now briefly argue how our propositions can benefit future computing. We noted that tomorrow's compute world is faced with numerous new applications that try to solve complex problems whose behaviors might not be fully understood and coded statically. Such applications require the means that can detect these behaviors dynamically. For example, MM5 [8] is a well-known traditional weather forecasting model that uses a very restricted number of weather parameters such as temperature and humidity, and takes the integral of a formulation of these parameters to predict weather in a given meshed geographical region. To improve the accuracy and geographical extent of predictions, more advanced models such as the WRF model [9] have been introduced. Interestingly, WRF uses the same algorithms as in MM5 for small size regions, a la taking the integral of a number of weather parameters though larger in numbers than in traditional MM5, but unexpectedly changes its behavior by taking derivations of parameters when forecasting weather for inter-regions that may be due to unexpected changes in the wave structure and energy. This is to say that the behaviors of processes comprising such an application are at the least very complex and hard to predict statically, implying that the pattern of their requests for compute resources are also hard to predict.

The old order-oriented approaches to management of such processes take one of the following actions in response to requests of a process for compute resources (i.e., CPU, I/O, file, and memory):

1. Blindly satisfy or reject the request of each process solely based on the local availability or unavailability of the requested resource irrespective of previous pattern of resource requests of this process and irrespective of the role and relation of this process to other processes in the community of this process that together make up the process population of an application. We believe this kind of ah-hoc response by process manager to process requests cannot satisfy many quality requirements of future computing applications such as high performance requirement.
2. Decide how to handle the request based on compile-time analysis of process requests. This is restricted to applications whose resource request patterns of processes of an application can be fully determined at compile-time; the very condition that is hardly satisfied in future complex applications such as the stated WRF weather forecasting example.
3. Deploy an additional component in the operating system (process manager) to profile the pattern of resource requests by the process through repetitive runs of the application to which the process belongs, and use this pattern to handle the request accordingly. Apart from the high run-time overhead of profiling, the addition of an extra component in the operating system for this purpose contradicts with the principle of keeping the operating system as primitive as possible. Furthermore, as in the first ad-hoc alternative, the decision on how to handle the request of the process is made irrespective of the role and relation of this process to other processes in its community that together make up the process population of an application.

The proposed needs-oriented approach to management of processes comprising a complex application uses the proposed category of process needs to avoid the shortcomings of the traditional order-oriented approaches. It achieves this by deciding on how to handle requests of a process based on current needs status of the process at run-time considering the patterns of resource requests of all other processes in the community of this process together forming an application. It does not require repetitive runs of application and saves us from adding a profiler to the operating system too. Nevertheless, how can this be done?

Without entering into implementation details, we suffice to answer this question by presenting our first intuition on how we may implement this kind of process management.

We know that the operating system keeps all types of information on processes in its own data structures in the kernel space. The process manager can share this information on a process with the process before deciding how to handle the request of the process. This reflection allows the process to request more consciously according to its hierarchy of needs (as proposed before). To give an example, a process that finds out that its vital needs have not been satisfied yet, stops asking for transcendence needs in vein.

On the other hand, the process manager can look up at current values in data structures storing information on all processes of an application and find out about the needs states of processes to determine if it had to upgrade the level of needs status of any process or not. It can also get a clue on future resource-request patterns of processes and accordingly

 ISBN: 978-1-61208-154-0

reconfigure itself dynamically. For example, it can kill the random identity generator process that generates unique system-wide identifiers for new forked processes in the application if it expects that the application will not fork new processes anymore. It may start working out how to deploy a remote resource if it finds out that a process will ask for the resource soon but it knows that this resource is not available locally. Overall, processes submit requests less frequently more sensibly and more orderly, process manager is more aware of request patterns and needs status of processes allowing it to be more responsive and respective to reasonable requests of processes representing the community of an application at run-time. This all leads to a much better well structured and well behaved process society that can handle the complex and extremely dynamic applications of the future.

V. CONCLUSION

Arguing against the shortcomings of traditional process managers for proper handling of process requests, and by using the Maslow's school of human management, we proposed a 5-layered pyramid of process needs that local and global managers of the process society, namely the computer operating systems of the future, must administer. We also drafted the responsibility of process managers based on the given categorized needs. We think conscious processes alongside process-aware process managers of the kind presented in this paper build up a more promising and manageable computing environment that has relatively large-scale orders of magnitudes of processes and resources distributed in wide scales.

We are currently working on a prototyped implementation of a Linux-based process manager enhanced with the proposed mechanism in this paper to provide process consciousness in terms of categorized process needs.

REFERENCES

[1] A. H. Maslow, Maslow on Management, Wiley, Ed.1, 1998.

[2] A. Adler, Social Interest, Faber & Faber, London, 1938.

[3] D. O'Connor and L. Yballe, "Maslow revisited: constructing a road map of human nature", *Journal of Management Education*, vol. 31, no. 6, pp. 738-756, 2007.

[4] P. F. Drucker, Management Challenges for the 21st Century, Butterworth-Heinemann, UK, 2006.

[5] E. L. Deci and R. M. Ryan, "The what and why of goal pursuits: human needs and the self-determination of behavior", *Psychological Inquiry*, vol. 11, no. 4, Page 227, 2000.

[6] J. J. O'Toole and K. J. Meier, "The human side of public organizations contributions to organizational performance laurence", *The American Review of Public Administration*, vol. 39, no. 5, pp. 499-518, 2009.

[7] P. Lawrence and N. Nohria, Driven: How Human Nature Shapes Organizations, Harvard Business School Working Knowledge, 2001.

[8] Ó. Rögnvaldsson, J. W. Bao, H. Ágústsson1 and H. Ólafsson, "Downslope windstorm in Iceland – WRF/MM5 model comparison", *Atmos. Chem. Phys.*, vol. 11, pp. 103–120, 2011.

[9] R. O. Olatinwo, T. Prabha1, J. O. Paz1, D. G. Riley and G. Hoogenboom, "The Weather Research and Forecasting (WRF) model: application in prediction of TSWV-Vectors Populations", *Journal of Applied Entomology*, vol. 135, no. 1-2, pp. 81–90, 2011

 ISBN: 978-1-61208-154-0

Parallel Processing for 3-D Slope Stability Problems

Cheng Yung Ming
Department of Civil and Structural Engineering
Hong Kong Polytechnic University, Hong Kong
Hong Kong, China

Li Na
Department of Civil and Structural Engineering
Hong Kong Polytechnic University, Hong Kong
Hong Kong, China

Abstract—**Rigorous three-dimensional (3D) slope stability analysis is time consuming and takes major computer resources. To speed up the computation so that engineers can get results within shorter time is a bottleneck problem. In this study, we propose to speed up 3D slope stability analysis program with "OpenMP" which is tested to be effective, by using parallel processing. Further, the authors demonstrate that calculation time by using OpenMP becomes much less than before and this kind of parallel processing is not difficult to be used for general problems. Through the analysis, we have also discovered the number of computers' CPU has a great impact on the performance of the 3-D slope stability program. There are various precautions in using parallel processing, or else there will be conflict of memory address and changes of values by different threads of processes.**

Keywords-parallel processing; limit equilibrium method; OpenMP; factor of safety.

I. INTRODUCTION

Landslip is one of the most serious disasters all over the world. To assess the potential dangerous of slopes, different slope stability analysis methods were developed and improved by researchers and engineers in the past. Among these methods, Limit Equilibrium Method (LEM) has been considered much by researchers in slope stability analysis.

All slope failures are three-dimensional (3-D) in nature. However, 3-D analysis based on limit equilibrium method is still seldom adopted in practice at present, because of the following limitations:

1. Direction of sliding is not considered in most existing slope stability formulations so that the problems under consideration must be symmetrical in geometry and loading;
2. Location of critical nonspherical 3-D failure surface under conditions is a difficult global optimization problem which has not been solved effectively; and
3. Existing methods of analyses are numerically unstable under transverse horizontal forces.

Two dimensional (2-D) modeling is usually adopted to greatly simplify the problem. However, all the actors of afety (FOS) obtained by 2-D slope stability methods are the approximate values, even though within a short computation time. While 3-D LEM can obtain a more reliable FOS with less assumptions, and more accurate consideration on the shape of the slope, but the computation time are highly increased at the same time.

Moreover, computer techniques are improved very fast in the past few decades and multiple CPUs/memory computers are largely applied in computing and engineering technology nowadays. It is a good chance for geotechnical engineers to perform parallel processing technique into slope stability analysis program to shorten the running time of 3-D slope stability programs.

In this research, parallel processing is proposed, and OpenMP, a shared-memory application programming interface (API) was tried. Calculation time, the bottleneck problem confused by most of the civil engineers, is testified to be shortened in 3-D limit equilibrium analysis in this study. There are seldom 3-D slope stability programs applying parallel processing recently. So it is a breakthrough in calculation time efficiency in 3-D slope stability analysis.

In this article, 3-D LEM methods are presented first, then the definition of parallel processing and OpenMP are briefly described. The main part of this research is to speed up for "SLOPE3D" by OpenMP, the authors explain how to do parallel processing in 3-D slope stability program, and the speedup efficiency is also discussed.

II. THREE-DIMENSIONAL LIMIT EQUILIBRIUM METHOD

The use of three-dimensional slope stability method becomes necessary as increasing requirement for accuracy in slope stability analysis.

Baligh and Azzouz [3], Azzouz and Baligh [1] presented a method that extends the concept of the 2-D circular arc shear failure method to 3-D slope stability problems. The method was just appropriate for a slope in cohesive soil. The results obtained by the method showed that the 3-D effects could lead to 4 to 40 percent increase in the factor of safety.

Chen and Chameau [6] extended Spencer's 2-D method to 3-D. The sliding mass was assumed to be symmetrical and divided into several vertical columns. The inter-column forces had the same inclination throughout the mass, and the shear forces were parallel to the base of the column.

Hunger [10] proposed a 3-D method that is a direct extension of the assumptions associated with Bishop's [5] 2-D simplified method. He assumed that the vertical shear forces acting on both longitudinal and the lateral vertical faces of each column can be neglected in the equilibrium equations. Also the vertical force equilibrium equation of each column and the summary moment equilibrium equation of the entire assemblage of columns are sufficient conditions to determine all the unknown forces.

Cheng [7] proposed a new formulation for 3-D Morgenstern-Price Method. The sliding mass was divided into several vertical columns and three assumptions are

 ISBN: 978-1-61208-154-0

included in the formulation: (1) Mohr–Coulomb failure criterion is valid; (2) For Morgenstern–Price's method, the factor of safety is determined based on the sliding direction where factors of safety with respect to force and moment are equal; (3) Sliding direction is the same for all soil columns (a unique sliding direction). The Morgenstern-Price formulation can also be simplified to 3-D Bishop and Janbu's simplified method by only consider force or moment equilibrium equation and neglecting all the inter-column vertical and horizontal shear force.

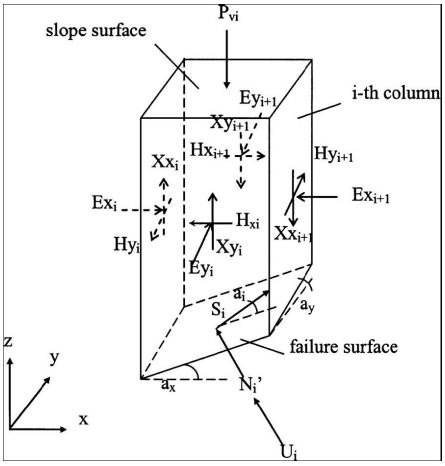


Figure 1. External and internal force acting on a typical soil column

A unique sliding direction shown in Figure 1 can be determined in the present 3D slope stability methods, which can also prevent the spread of the sliding when considering transverse earthquake load. And this method can easily converge. (Cheng, [8])

III. PARALLEL COMPUTERS – OPENMP PROGRAM

Today's computer systems become highly complex. Multiple CPUs and memory appear. As a result, a computer might be able to fetch a datum from memory, multiply two floating-point numbers, and evaluate a branch condition at the same time. We call this kind of function parallel processing and those computers are shared-memory parallel computers or multiprocessor computers (Chapman, [4]).

Although modern computers can achieve parallel processing, it's impossible to process automatically. Therefore compilers are necessary to identify independent streams of instructions which can be executed in parallel and OpenMP is one of the programming interfaces which can help us to do parallel processing by adding it in the sequential program.

OpenMP is a shared-memory application programming interface (API) but it is not a new programming language. It can be coded by FORTRAN (most of the civil program is written by FORTRAN), C and C++ to describe how work is to be shared among threads which will execute on different processors or cores and to order access for sharing data as needed. OpenMP supports the so-called fork-join programming model illustrated in Figure 2. Under this approach, the program starts as a single thread (initial tread) of execution, just like a sequential program. Once an OpenMp parallel is constructed, the program is executed automatically to create a team of threads (Fork) implementing independent work and only the original thread, or master of the team is continuous at the end of the construction (Join). The space between Fork and Join is called a parallel region. (Chapman, Jost, 2008)

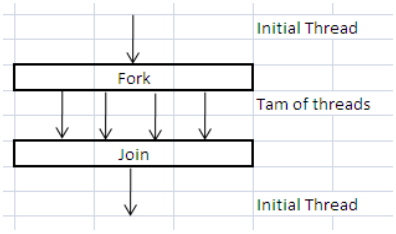


Figure 2. The Fork-join programming mode support by OpenMP

OpenMP have two major advantages, one is its strong emphasis on structured parallel programming and the other is simples to use. Those reasons make it popular and run on many different platforms nowadays (Chapman, [4]).

IV. SPEEDING UP FOR 3D SLOPE STABILITY ANALYSIS PROGRAM BY "OPENMP"

Due to the large number of unknown and complicated calculation procedure in three-dimension LEM, it is very time consuming to calculate minimum factor of safety with high accuracy for a wide range of possible failure surface with a large scale and complicate slope (usually with more than two soil layers) by 3-D "rigorous" limit equilibrium method. And the calculation time can be counted in number of days.

Parallel processing by shared-memory parallel computers or multiprocessor computers is proposed in the paper to shorten the computation time for three-dimensional limit equilibrium method using OpenMP. Slope stability analysis program "SLOPE3D" is a sequential program written in FORTRAN and was speeded up by OpenMP in this research. The methods and speedup efficiency of using OpenMp in "SLOPE3D" are discussed in the following sections. A 3-D slope model case of "SLOPE3D" is shown in Figure 3.

A. How to Use OpenMP in Three-Dimensional Slope Stability Analysis Program

First step is to classify the part of the program which is suitable to implement parallelization to gain the maximum efficiency. However, it's quite difficult to parallel all parts of the program. The procedure of 3D slope stability analysis program can be mainly distributed into four parts, input reading, previous calculation (usually for simple calculations which can obtain object values in main calculation by changing the input value), main calculation and output generation. The main calculation occupies large part of computational procedures and it takes most of the computation time, so parallelization was tried in this part to obtain the maximum efficiency. Two ways of parallel

 ISBN: 978-1-61208-154-0

processing are tried in this research to use in the main calculation part of "SLOPE3D", illustrated in Figure 4 and 5.

In Model 1 of Figure 4, two parallel regions are generated and a new matrix value Temp_TTX(i,j) is developed in 1st region to store all the initial calculating values. Open MP enables program to create team of threads to distribute the Do-Loops according to the number of CPUs, e.g., 4 CPUs, 4 parts of Do-Loops. In 2nd region, all threads will simultaneously read the values stored in TTX, add their value in Temp_TTX (i,j), and write out the results in the single storage location for summation TTX.

Different from Model 1, there was only one parallel region in Model 2 in Figure 5. Both values of Temp_TTX and summation values of TTX are calculated in the same parallel region. Each thread (CPU) will make a copy for the value; read and write will only execute within the thread. 4 different values of Temp_TTX will be stored in each thread (CPU). To prevent "Data Competition", a Critical Clause is added to allow only one thread to do calculation to obtain summation values of TTX from 4 private Temp_TTX. Model 2 reduces the number of parallel regions from 2 to 1, which can reduce the fork and join time.

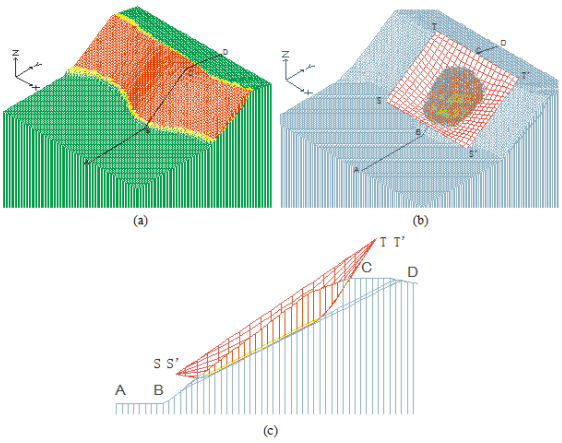


Figure 3. "Slope 3D" model of a concave and convex slope combination: (a) a concave and convex slope highlighted in red region; (b) FEM meshing made in the sliding area; (c) cross-section of the failure surface of the slope

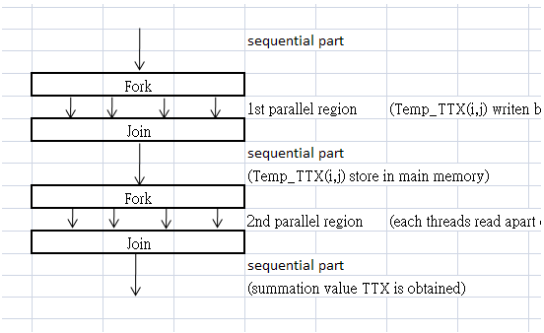


Figure 4. Mine-Map of parallel processing Model 1

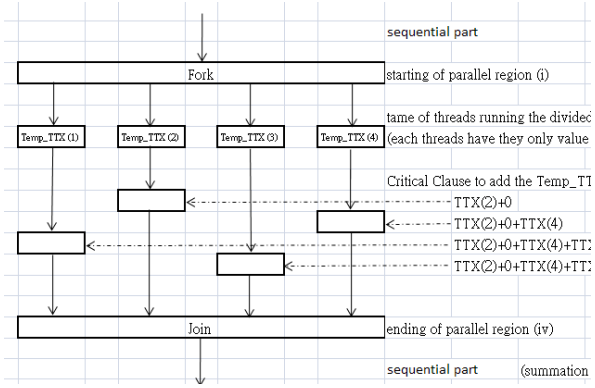


Figure 5. Mine-Map of parallel processing Model 2

B. *Running Time Comparison*

In order to check the efficiency of those parallel processing models above, a traditional 3-D slope is analyzed by the parallel programs mentioned above with 3-D Morgenstern-Price Method. Two issues are focused on the comparison; one is the running time against number of divided soil columns. The number of soil columns divided in failure mass of slope is an important factor affecting the accuracy of factor of safety. The other issue is the running time of computers with different number of CPUs. The results are shown in Table 1, 2 and 3 with 2 CPUs, 4 CPUs and 8 CPUs computer, respectively.

Several issues can be observed from the results shown below. First, for the third lines in the nethermost part of these three curves which is the running time for original sequential program are quite linear in Figure 7 and 8. It can be assumed that the running time of sequential 3-D slope analysis program is directly proportional to the number of soil column divided in the same slope analysis problem.

Moreover, there are apparent differences on running time between the original program and the two parallel processing programs when the numbers of soil columns are higher than 100,000. Also the difference is increasing when the numbers of divided soil columns are increasing. This is reasonable, because in respect to the typical calculation element, the increasing numbers of soil columns are referring to the size of the Do-Loop. Therefore, the calculation time for the Do-Loop will be increase and the performances of parallel processing should also be enhanced.

Although the difference between the running time in two parallel processing program are not large, the performance of program using parallel processing Model 2 can be noticed being better than Model 1, for the reason that extra running time can be reduced by decreasing the number of parallel regions and the matrix value Temp_TTX(i,j) shown in Figure 5.

On the other hand, if we focus on the speedup percentage in these three groups of results, negative value is found when the numbers of soil columns are lower than 40,000. This can be explained by the time using in forking and joining the paralleled threads. OpenMP supports the so-called fork-join programming model, illustrated in Figure 2, and extra

 ISBN: 978-1-61208-154-0

programming time is used to this model. If the reduced calculation time we gain due to parallel processing are lower than the forking and joining time we use for the paralleled threads, negative speedup will occur. In order to obtain a positive speedup percentage, a boundary can be set to limit the use of parallel processing which will only occur when the number of divided are higher than 40,000.

The maximum speedup percentage for 2 CPUs, 4CPUs and 8 CPUs computer are 27.31%, 29.95% and 36.5%, respectively. Higher speedup percentage can be obtained for computer with more CPUs. This is because, for the same size of original Do-Loop in the program, more Do-Loops with relative smaller size can be divided when more number of threads (number of CPUs) is provided. Much running time can be reduced due to the Do-Loops can run in once time.

TABLE I. RUNNING RESULT FROM 3-D MORGENSTERN-PRICE METHOD IN 2 CPUS COMPUTER

Analysis method :		**Morgenstern-Price's Method**				
No. of Cycles :		100				
No. of CPU		2				
Spacing	**No. of soil columns**	**Original running time (s)**	**OpenMP running time (s)**		**Speed Up (%)**	
			OMP 1	OMP 2	OMP 1	OMP 2
0.5	1,600	8.391	21.297	16.922	-153.81	-101.67
0.4	2,500	12.75	24.985	20.453	-95.96	-60.42
0.25	6,400	32.969	42.547	38.125	-29.05	-15.64
0.2	10,000	51.781	59.969	54.906	-15.81	-6.04
0.1	40,000	288.641	260.656	246.734	9.70	14.52
0.05	160,000	1561.656	1327.985	1307.937	14.96	16.25
0.04	250,000	2637.532	2196.875	2028.203	16.71	23.10
0.03	443,556	4819.532	3805.922	3503.328	21.03	27.31
0.025	640,000	11352.75	9282.125	8642.453	18.24	23.87

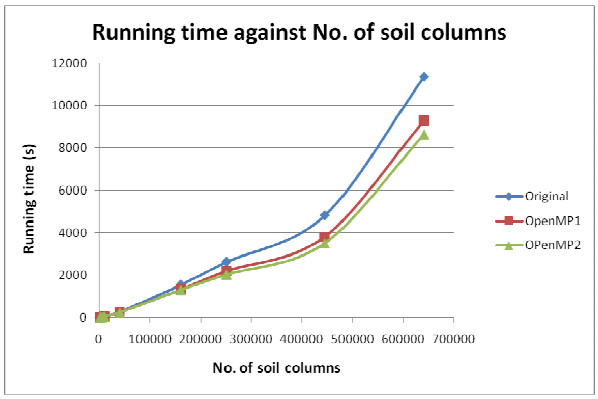


Figure 6. Running time against number of divided column in 2 CPUs computer.

TABLE II. RUNNING RESULT FROM 3-D MORGENSTERN-PRICE METHOD IN 4 CPUS COMPUTER

Analysis method :		**Morgenstern-Price's Method**				
No. of Cycles :		100				
No. of CPU		4				
Spacing	**No. of soil columns**	**Original running time (s)**	**OpenMp running time (s)**		**Speed Up (%)**	
			OMP 1	OMP 2	OMP 1	OMP 2
0.5	1,600	12.67	26.875	24.312	-112.12	-91.89
0.4	2,500	17.97	44.312	28.516	-146.59	-58.69
0.25	6,400	47.58	57.203	51.484	-20.22	-8.21
0.2	10,000	73.56	76.891	71.906	-4.53	2.25
0.1	40,000	338.7	293.641	280.172	13.30	17.28
0.05	160,000	1472.52	1181.703	1107.156	19.75	24.81
0.04	250,000	2416.39	1829.391	1838.923	24.29	23.90
0.03	443,556	4069.52	3044.063	2850.516	25.20	29.95
0.025	640,000	6082.75	4909.578	4743.078	19.29	22.02

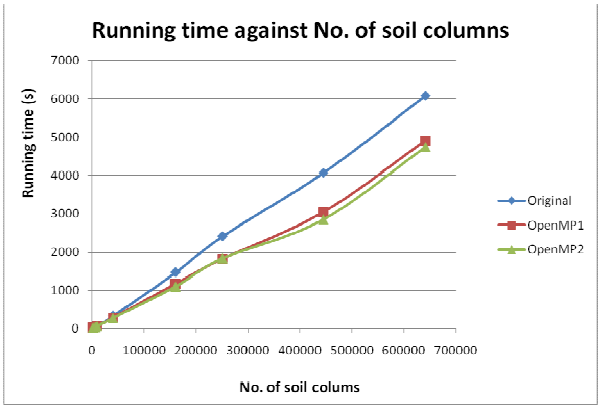


Figure 7. Running time against number of divided column in 4 CPUs computer.

TABLE III. RUNNING RESULT FROM 3-D MORGENSTERN-PRICE METHOD IN 8 CPUS COMPUTER

Analysis method :			**Morgenstern-Price's Method**			
No. of Cycles :			100			
No. of CPU			8			
Spacing	**No. of soil columns**	**Original running time (s)**	**OpenMP running time (s)**		**Speed Up (%)**	
			OMP 1	**OMP 2**	**OMP 1**	**OMP 2**
0.5	1,600	8.094	39.297	33.609	-385.51	-315.23
0.4	2,500	11.922	42.5	36.453	-256.48	-205.76
0.25	6,400	33.5	59.063	53.203	-76.31	-58.81
0.2	10,000	51.765	74.469	71.89	-43.86	-38.88
0.1	40,000	247.313	227.906	214.7973	7.85	13.15
0.05	160,000	1063.859	843.578	835.391	20.71	21.48
0.04	250,000	1803.406	1298.266	1308.688	28.01	27.43
0.03	443,556	3229.969	2108.297	2050.922	34.73	36.50
0.025	640,000	4854.312	3502.86	3265.375	27.84	32.73

 ISBN: 978-1-61208-154-0

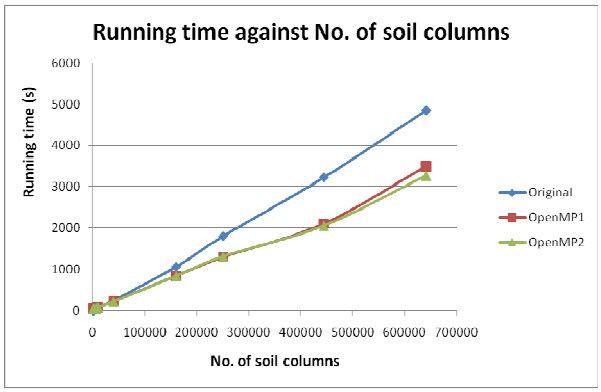


Figure 8. Running time against number of divided column in 8 CPUs computer.

V. CONCLUSIONS AND DISCUSSIONS

Based on the present study, the theoretical backgrounds for 3-D limit equilibrium methods and OpenMP were reviewed. The basic concept of parallel processing and OpenMP are also studied above. Furthermore, how to do parallel processing in 3-D limit equilibrium program "SLOPE3D" is described and its efficiency is also discussed. The following conclusions are summarized and presented for parallel processing in slope stability program:

- OpenMP, a shared-memory application programming interface (API) is a suitable interface for civil engineers to do parallel processing in slope analysis program to reduce the calculation time. It is relatively easy for use and can be added into a sequential program coded in FORTRAN.
- The efficiency of using OpenMP is highly affected by the reduction time obtained from paralleled Do-Loop and the forking & joining time used in the teams of thread.
- About one-third of the calculation time can be reduced by using OpenMP and the larger numbers of CPUs exist in the analyzed computer, the better performance of the program is gained.

Comparing to other shared-memory programming interfaces, OpenMP has better compatibility than them. Take MPI as an example, for those researchers and engineers who do not have enough computer skills, OpenMp not only can be easily used to solve general problems, but also runs well in every computer without modifying the code, while MPI code can only face one single server. Moreover, using the server with higher number of processors to speed up the efficiency of numerical program is not widely applied in Civil engineering. And the 30% time-shortening implementation in this article is acceptable to geotechnical engineers. There is no need to spend so much time on a shared-memory programming interface as complicated as MPI.

At future work, the specific code for the application can be further parallelized. The authors will consider trying some different shared-memory API, further parallelizing the specific code, and experimenting with a higher size experiment with higher number of processors if the equipment is available.

REFERENCES

[1] Anderson, M. G. and Richards, K. S. (1987), Slope Stability: Geotechnical Engineering and Geomorphology, John Wiley and Sons, N.Y.

[2] Azzouz, A. S. and Baligh, M. M. (1978). Discussion on Three-dimensional slope stability analysis method. J. Geotech. Engng Div. ASCE, 104,GT9, 1206-1208.

[3] Baligh, M. M., and Azzouz, A. S. (1975). End effects on stability of cohesive slopes. J. Geotech. Engrg. Div., ASCE, 101(11), 1105–1117.

[4] Chapman B., Jost G., and Ruud v. d. Pas (2008), Using OpenMP : portable shared memory parallel programming, Cambridge, Mass. : MIT Press, pp. 1-3, 8-24

[5] Bishop, A. W. (1955). The use of the slip circle in the stability analysis of slopes. Ge´otechnique, London, V(1), 7–17.

[6] Chen, R. H. and Chameau, J. L. (1983), Three-dimensional limit equilibrium analysis of slopes, Geotechnique, 33(1), pp. 31-40.

[7] Cheng Y.M. and C. J. Yip (2007), Three-Dimensional Asymmetrical Slope Stability Analysis Extension of Bishop's, Janbu's, and Morgenstern–Price's Techniques, Journal of Geotechnical and Geoenvironmental Engineering, ASCE ,Vol. 133, No. 12, December 2007, pp. 1544-1555

[8] Cheng Y.M. and C.K. Lau (2008), Slope stability analysis and stabilization: new methods and insight , Routledge Publishers, pp. 17-21

[9] Fredlund, D. G., and Krahn, J. (1977), Comparison of Slope Stability Methods of Analysis, Canadian Geotechnical Journal, 14, pp. 429-439

[10] Hungr, O. (1987), An extension of Bishop' Simplified Method of slope stability analysis to three dimensions, Geotechnique, Vol. 37, No. 1, pp. 113-117

[11] Chandra R., Dagum L., Kohr D., Maydan D., McDonld J., and Menon R. (2001), Parallel programming in OpenMP, Morgan Kaufmann Publishers, pp. 33-35

 ISBN: 978-1-61208-154-0

Agent-Oriented Computing: Agents as a Paradigm for Computer Programming and Software Development

Alessandro Ricci
DEIS, Alma Mater Studiorum – Università di Bologna
Via Venezia 52, 47521 Cesena (FC), Italy
a.ricci@unibo.it

Andrea Santi
DEIS, Alma Mater Studiorum – Università di Bologna
Via Venezia 52, 47521 Cesena (FC), Italy
a.santi@unibo.it

***Abstract*—The notion of *agent* more and more appears in different contexts of computer science, often with different meanings. The main acceptation is the AI (Artificial Intelligence) and Distributed AI one, where agents are essentially exploited as a technique to develop special-purpose systems exhibiting some kind of intelligent behavior. In this paper, we introduce a further perspective, shifting the focus from AI to computer programming and programming languages. In particular, we consider agents and related concepts as general-purpose abstractions useful for programming software systems in general, conceptually extending object-oriented programming with features that – we argue – are effective to tackle some main challenges of modern software development. Accordingly, the main contribution of the work is first the definition of a conceptual space framing the basic features that characterize the agent-oriented approach as a programming paradigm, then its validation in practice by using a platform called JaCa, with real-word programming examples.**

Keywords-agent-oriented programming; multi-agent systems; concurrent programming; distributed programming

I. INTRODUCTION

The notion of *agent* more and more appears in different contexts of computer science, often with different meanings. In the context of Artificial Intelligence (AI) or Distributed AI, agents and multi-agent systems are typically exploited as a technique to tackle complex problems and develop intelligent software systems [16][32][27]. In this paper, we discuss a further perspective, which aims at exploiting agents and agent-oriented abstractions to devise a high-level computing programming paradigm for developing software, as a natural evolution of objects (as defined in OOP) and actors [6]. So, instead of exploiting agents as abstractions to support AI techniques, here we frame the value of multi-agent programming as a general-purpose paradigm for organizing and programming software, providing features that we consider effective to tackle main challenges of modern and future software development, such as concurrency, decentralization of control, distribution, autonomy, adaptivity.

Concurrency, in particular, due to the spread of multi-core technologies, is more and more a core issue of mainstream programming—besides the academic research contexts where it has been studied for the last fifty years. This situation is pretty well summarized by the sentence: "The free lunch is over" as put by Sutter and Larus in [30]. Besides introducing fine-grain mechanisms or patterns to exploit parallel hardware and improve the efficiency of programs in existing mainstream languages, it is now increasingly important to introduce higher-level abstractions that "help build concurrent programs, just as object-oriented abstractions help build large component-based programs" [30]. We argue that agent-oriented programming – as framed in this paper – provides one such level of abstraction. Besides concurrency, we believe that the level of abstraction introduced by an agent-oriented programming paradigm would be effective to tackle the complexities introduced by modern and future application domains, such as cloud computing, autonomic computing, pervasive computing and so on.

Actually, the idea of Agent-Oriented Programming is not new. The first paper about AOP is dated 1993 [28], and since then many Agent Programming Languages (APL) and languages for Multi-Agent Programming have been proposed in literature [11][12][13]. The objective of AOP as introduced in [28] was the definition of a post-OO programming paradigm for developing complex applications, providing higher-level features compared to existing paradigms. In spite of this objective, it is apparent that agent-oriented programming has not had a significant impact on mainstream research in programming languages and software development, so far. We argue that this depends on the fact that (in spite of few exceptions) most of the effort and emphasis have been put on theoretical issues related to AI themes, instead of focusing on the key principles and practice of general-purpose computer programming. This is the direction that we aim at exploring in our work and in this paper.

The remainder of the paper is organized as follows. After presenting related works (Section II), we first define a conceptual space to describe the basic features of a general-purpose programming paradigm based on agent-oriented abstractions (Section III). Then, we provide a first practical evaluation by exploiting an agent-oriented platform called JaCa (Section IV), which actually integrates two different existing agent technologies, Jason [9][10] and

 ISBN: 978-1-61208-154-0

CArtAgO [25]. The objective is to show how to exploit agent-oriented abstractions to conceive and develop real-world programs, and point out outcomes and limitations of current models and technology. Finally we close the paper with some concluding remarks (Section V).

II. Related Works

Most of the agent-oriented programming languages and technologies – in particular those based on high-level computational model/architecture such as the BDI (Belief-Desire-Intention) one [23] – have been introduced in (Distributed) Artificial Intelligence, so targeted to problems in that context [11][12][13]. Besides this main perspective, in the context of AOSE (Agent Oriented Software Engineering) some agent-oriented *frameworks* based on mainstream programming languages – such as Java – have been introduced, targeted to the development of complex distributed software systems. A main example is JADE (Java Agent DEvelopment Framework) [8], a FIPA-compliant [1] platform that makes it possible to implement multi-agent systems in Java. JADE is based on a weak notion of agency: JADE agents are Java-based actor-like active entities, communicating by exchanging messages based on FIPA ACL (Agent Communication Language). So there is not an explicit account for high-level agent concepts – goals, beliefs, plans, intentions are examples, referring to the BDI model – that are exploited instead in agent-oriented programming languages to raise the level of abstraction adopted to define agent behaviour. Also, JADE has not an explicit notion of agent environment, defining agent actions and perceptions, which are key concepts for defining agent reactiveness. Differently from JADE, the JaCa platform presented in this paper allows for programming agents using a BDI-based computational model and has explicit notion of shared programmable environments – perceived and acted upon by agents – based on the A&A (Agents and Artifacts) conceptual model [20], described in next sections.

Another example of Java-based agent-oriented framework is simpA [26], which has been conceived to investigate the use of agent-oriented abstractions for simplifying the development of concurrent applications. simpA shares many points with the perspective depicted in this paper: however it is based in on a weak model of agency, similar to the one adopted in JADE. Differently from JADE, it explicitly supports a notion of environment, based on A&A.

Besides the different underlying computational models, both JADE and simpA do not explicitly introduce a new full-fledge agent-oriented programming language for programming agents, being still based on Java. A different approach is adopted by JACK [15], a further platform for developing agent-based software which *extends* the Java language with BDI constructs – such as goals and plans – for programming agents, integrating the object-oriented and agent-oriented levels. Finally, similarly to JADE, Jadex [22] is a FIPA compliant framework based on Java and XML, but adopting the BDI as underlying agent architecture.

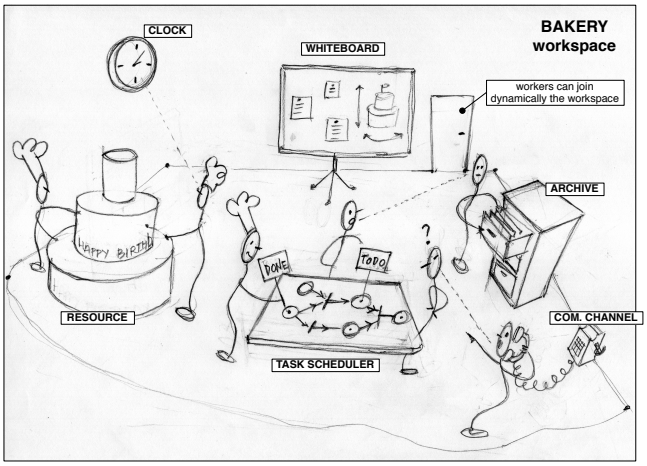


Figure 1. Abstract representation of the A&A metaphor in the context of a bakery.

III. Agent-Oriented Abstractions for Computer Programming

Quoting Lieberman [18], *"The history of Object-Oriented Programming can be interpreted as a continuing quest to capture the notion of abstraction – to create computational artifacts that represent the essential nature of a situation, and to ignore irrelevant details"*. In that perspective, in this section we identify and discuss a core set of concepts and abstractions introduced by agent-oriented programming. While most of these concepts already appeared in literature in different contexts, our aim here is to highlight their value for framing a conceptual space and an abstraction layer useful for defining general-purpose programming languages.

A. The Background Metaphor

Metaphors play a key role in computer science, as means for constructing new concepts and terminology [31]. In the case of objects in OOP, the metaphor is about real-world objects. Like physical objects, objects in OOP can have properties and states, and like social objects, they can communicate as well as respond to communications. In the case of actors [6], similarly, the inspiration is clearly more anthropomorphic, and a variety of anthropomorphic metaphors influenced its development [29][17].

The inspiration for the agent-oriented abstraction layer that we discuss in this paper is anthropomorphic too and refers to the A&A (Agents and Artifacts) conceptual model [20], which takes human organizations as main reference. Figure 1 shows an example of such metaphor, represented by a human working environment, a *bakery* in particular. It is a system where articulated concurrent and coordinated activities take place, distributed in time and space, by people working inside a common environment. Activities are explicitly targeted to some objectives. The complexity

 ISBN: 978-1-61208-154-0

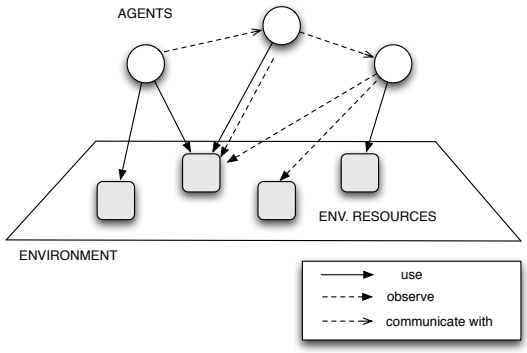


Figure 2. Abstract representation of an agent-oriented program composed by agents working within an environment.

of work calls for some division of labor, so each person is responsible for the fulfillment of one or multiple tasks. Interaction is a main dimension, due to the dependencies among the activities. Cooperation occurs by means of both direct verbal communication and through tools available in the environment (e.g., a blackboard, a clock, the task scheduler). So the environment – as the set of tools and resources used by people to work – plays a key role in performing tasks efficiently. Besides tools, the environment hosts resources that represent the co-constructed results of people work (e.g., the cake).

Following this metaphor, we see a program – or software system – as a collection of autonomous agents working and cooperating in a shared environment Figure 2: on the one side, agents (like humans) are used to represent and modularize those parts of the system that need some level of autonomy and pro-activity—i.e., those parts in charge to autonomously accomplish the tasks in which the overall labor is split; on the other side, the environment is used to represent and modularize the non-autonomous functionalities that can be dynamically composed, adapted and used (by the agents) to perform the tasks.

A main feature of this approach is that it promotes a *decentralized mindset* in programming, as also considered by Resnick in [24]. Such a mindset has two main cornerstones.

The first one is the *decentralization and encapsulation of control*: there is not a unique locus of control in the system, which is instead decentralized into agents. It is worth remarking that here we are assuming a logical point of view over decentralization—not strictly related to, for instance, physical threads or processes. The agent abstraction extends the basic encapsulation of state and behavior featured by objects by including also encapsulation of control, which is fundamental for defining and realising agent *autonomous* behaviour.

The second cornerstone is the interaction dimension which includes coordination and cooperation. There are two basic orthogonal ways of interacting: direct communication among agents based on high-level asynchronous message passing and environment-mediated interaction (discussed in Subsection III-D) exploiting the functionalities provided by environment resources.

B. Structuring Active Behaviors: Tasks and Plans

Decentralization and encapsulation of control, as well as direct communication based on message passing, are main properties also of actors, as defined in [6]. The actor model, however, does not provide further concepts useful to *structure* the autonomous behavior, besides a simple notion of *behavior*. This is an issue as soon as we consider the development of large or simply not naive active entities. To this end, the agent abstraction extends the actor one introducing further high-level notions that can be effectively exploited to organize agent autonomous behavior, namely *tasks* and *plans*.

The notion of task is introduced to specify a unit of work that has to be executed—the objective of agents' activities. So, an agent acts in order to perform a task, which can be possibly assigned dynamically. The same agent can be able to accomplish one or more types of task, and the *type* of the agent can be strictly related to the set of task types that it is able to perform.

Conceptually, an agent is hence a computing machine that, given the description of a task to execute, it repeatedly chooses and executes *actions* so as to accomplish that task. If the task concept is used as a way to define *what* has to be executed, the set of actions to be chosen and performed represents *how* to execute such tasks. The first-class concept used to represent one such set is the *plan*. So the agent programmer defines the behavior of an agent by writing down the plans that the agent can dynamically combine and exploit to perform tasks. For the same task, there could be multiple plans, related to different contextual conditions that can occur at runtime.

On the one side, tasks and plans can be used to define the contract explicitly stating what jobs the agent is able to do; on the other side, they are used (by the agent programmer) to structure and modularize the description of how the agent is able to do such jobs, organizing plans in sub-plans.

This approach makes it possible to frame a smooth path in defining different levels of abstraction in specifying plans and, correspondingly, different levels of autonomy of agents. At the base level, a plan can be a detailed description of the sequence of actions to execute. In this case, task execution is fully pre-defined, since the programmer is charged with the entire task specification; the level of autonomy of the agent is limited in selecting the plan among the possible ones specified by the programmer. In a slightly more complex case, a plan could be the description of a set of possible actions to perform, and the agent uses some criteria at runtime to select which one to execute. This enhances the level of autonomy of the agent with respect to what strictly specified by the programmer. An even stronger step towards

 ISBN: 978-1-61208-154-0

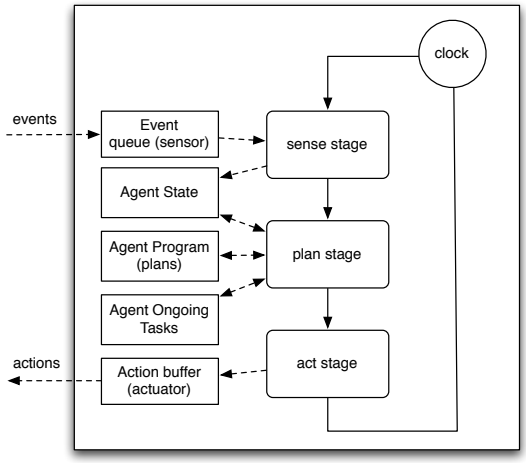


Figure 3. Conceptual representation of an agent architecture, with in evidence the stages of the execution cycle.

autonomy is given by the case in which a plan is just a partial description of the possible actions to execute, and the agent dynamically infers the missing ones by exploiting information about the ongoing tasks, and about the current knowledge of its state and the state of the environment.

C. Integrating Active and Reactive Behaviours: The Agent Execution Cycle

More and more the development of applications calls for flexibly integrating active and reactive computational behaviors, an issue which is strongly related to the problem of integrating thread-based and event-based architectures [14]. Active behaviors are typically mapped on OS threads, and the asynchronous suspension/stopping/control of thread execution in reaction to an event is an issue in high-level languages. So, for instance, in order to make a thread of control aware of the occurrence of some event – to be suspended or stopped – it is typically necessary to "pollute" its block of statements with multiple tests spread around.

In the case of agents, this aspect is tackled quite effectively by the control architecture that governs their execution, which can be considered both *event-driven* and *task-driven*. The execution is defined by a control loop composed by a possibly non-terminating sequence of execution cycles. Conceptually, an execution cycle is composed by three different stages (see Figure 3):

- *sense* stage – in this stage the internal state of the agent is updated with the *events* collected in the agent event queue. So this is the stage in which inputs generated by the environment during the previous execution cycle are fetched.
- *plan* stage – in this stage the next action to execute is chosen, based on the current state of the agent, the agent plans and agent ongoing tasks; additionally, agent state is also updated to reflect such a choice.
- *act* stage – in this stage the actions selected in the *plan* stage are executed.

The agent machine continuously executes these three stages, performing one execution cycle at each logical clock tick. Conceptually, the agent control flow is never blocked—actually it can be in idle state if, for instance, the executed plan states that no action has to be executed until a specific event is fetched in the sense stage. This architecture easily allows, for instance, for suspending a plan in execution and execute another plan to handle an event suddenly detected in the sense stage.

While in principle this makes an agent machine less efficient than machines without such loops, this architecture allows to have a specific point to balance efficiency and reactivity thanks to the opportunity to define proper atomic actions. Besides, in practice, by carefully design the execution cycle architecture, it is possible to minimize the overheads – for instance by avoiding to cycle and consuming CPU time if there aren't actions to be executed or new events to be processed – and eventually completely avoid overheads when needed —for instance, by defining the notion of atomic (not interruptible) plan, whose execution would be as fast as normal procedures or methods in traditional imperative languages.

D. "Something is Not an Agent": the Role of the Environment Abstraction

Often programming paradigms strive to provide a single abstraction to model every component of a system. This happens, for instance, in the case of actor-based approaches. In Erlang [7] for instance, which is actor-based, every macro-component of a concurrent system is a process, which is the actor counterpart. This has the merit of providing uniformity and simplicity, indeed. At the same time, the perspective in which everything is an active, autonomous entity is not always effective, at least from an abstraction point of view. For instance, it is not really natural to model as active entities either a shared bounded-buffer in producers/consumers architectures or a simple shared counter in a concurrent programs. In traditional thread-based systems such entities are designed as monitors, which are passive.

Switching to an agent abstraction layer, there is an apparent uniformity break due to the notion of *environment*, which is a first-class concept defining the context of agent tasks, shared among multiple agents.

From a designer and programmer point of view, the environment can be suitably framed as such non-autonomous part of the system which be used to encapsulate and modularize those functionalities and services that are eventually shared and exploited by the autonomous agents at runtime. More specifically, by recalling the human metaphor, the environment can be framed as the set of *resources* and *tools* that are possibly shared and *used* by agents to execute their

tasks. In that perspective, a bounded-buffer, a shared database etc. can be naturally designed and programmed as a shared resource populating the environment where – for instance – producers/consumers agents work.

E. Using and Observing the Environment

To be usable by agents, an environment resource provides a set of *operations* – that constitutes its *usage interface* – encapsulating some piece of functionality. Such operations are the basic actions that an agent can execute on instances of that resource type. So the set of actions that an agent can execute inside an environment depends on the set of resources that are available in that environment. Since resources can be created and disposed at runtime by agents, the agent action repertoire can change dynamically.

The execution of an operation (action) performed by an agent on a resource may complete with a success or a failure—so an explicit success/failure semantics is defined. Actions (operations) are performed by agents in the act stage of the execution cycle seen previously. Then, the completion of an action occurs asynchronously, and is perceived by the agent as a basic type of event, fetched in the sense stage. This can occur in the next execution cycle or in a future execution cycle, since the execution of an operation can be long-term. So, an important remark here is that the execution cycle of an agent *never blocks*, even in the case of executing actions that – to be completed – need the execution of further actions of other agents. This means that an agent, even if "waiting" for the completion of an action, can react to events perceived from the environment and execute a proper action, following what is specified in the plan.

Finally, aside to actions, *observable properties* and *observable events* represent the other side of agent-environment interaction, that is the way in which an agent gets input information from the environment. In particular, observable properties represent the observable state that an environment resource may expose, as part of its functionalities. The value of an observable property can be changed by the execution of operations of the same resource. A simple example is a counter, providing an inc operation (action) and an observable state given by an observable property called count, holding the current count value. By observing a resource, an agent automatically receives the updated value of its observable properties as percepts at each execution cycle, in the sense stage. Observable events represent possible signals generated by operation execution, used for making observable an information not regarding the resource state, but regarding a dynamic condition of the resource. Taking as a metaphor a coffee machine as environment resource, the display is an observable property, the beep emitted when the coffee is ready is an observable event. Choosing what to model as a property or as an event is a matter of environment design.

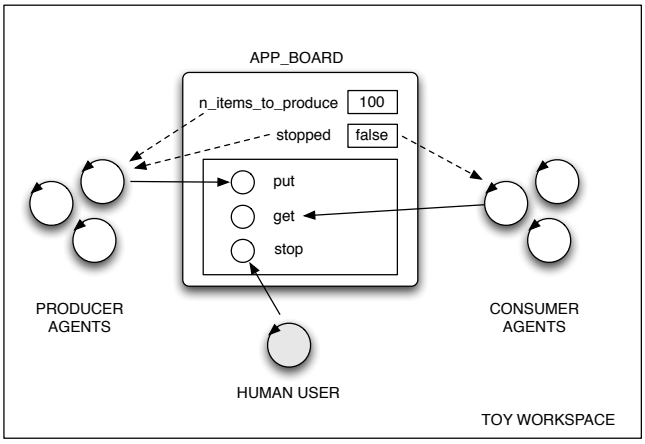


Figure 4. A toy workspace, with producer and consumer agents interacting by means of an app_board artifact.

IV. Evaluating the Idea with Existing Agent Technologies: The JaCa Platform

The aim of this section is to show more in practice some of the concepts described in the previous section. To this end, we will use existing agent technologies, in particular a platform called JaCa, which actually integrates two independent technologies: the Jason agent programming language [10] – for programming agents – and the CArtAgO framework [25], for programming the environment.

A. JaCa Overview

Following the basic idea discussed in Section III - a JaCa program is conceived as a dynamic set of autonomous agents working inside a shared environment, that they use, observe, adapt according to their tasks. The environment is composed by a dynamic set of environment resources which in CArtAgO are called "artifacts"—the term was inspired by Activity Theory and Distributed Cognition, where it is used to refer to any object that has been specifically designed to provide some functionality and which is used by humans to achieve their objective. Agents – by means of proper actions – can dynamically create and dispose artifacts, beside using them.

In the following, we introduce only those basic elements of agent and environment programming which are necessary to show the features discussed at the conceptual level in the previous section. To this end, we use a toy example which is about the implementation of a producers-consumers architecture, where a set of producer agents continuously and concurrently produce data items which must be consumed by consumer agents (see Figure 4). Further requirements – which make the example more interesting for our purposes – are that *(i)* the number of items to be produced is fixed, but the time for producing each item (by the different producers) is not known a priori; *(ii)* the overall process can be interrupted by the user anytime.

 ISBN: 978-1-61208-154-0

The task of producing items is divided upon multiple producer agents, acting concurrently—the same holds for consumer agents. To interact and coordinate the work, agents share and use an environment resource, the app_board artifact, which functions both as a buffer to collect items inserted by producers and to be removed by consumers and as a tool to control the overall process by the human user. The resource provides on the one side operations (actions for the agent) to insert (`put`), remove (`get`) items and to stop the overall activities (`stop`); on the other side, observable properties `n_items_to_produce` and `stopped`, keeping track of, respectively, the number of items still to be produced (which starts from an initial value and is decremented by the resource each time a new item is inserted) and the stop flag (initially false and set to true when the `stop` operation is executed).

In the following, first we give some glances about agent programming in Jason by discussing the implementation of a producer agent (see Table I), which must exhibit a pro-active behavior – performing cooperatively the production of items, up to the specified number – but also a reactive behavior: if the user stops the process, the agents must interrupt their activities. Then we briefly consider the implementation of the app_board artifact, to show in practice some elements of environment programming.

B. Programming Agents in Jason

Being inspired by the BDI (Beliefs-Desires-Intentions) architecture [23], the Jason language constructs that programmers can use can be separated into three main categories: *beliefs*, *goals* and *plans*. An agent program is defined by an initial set of *beliefs*, representing the agent's initial knowledge about the world, a set of *goals*, which corresponds to tasks as defined in Section III, and a set of *plans* that the agent can dynamically compose, instantiate and execute to achieve such goals.

In JaCa the beliefs of an agent can represent two types of knowledge:

- the agent internal state – an example is given by the `n_items_produced(N)` belief, which is used by a producer agent to keep track of the number of items produced so far;
- the observable state of the resources of the environment which the agent is observing – in the example, every producer agent observes the app_board artifact, which has two observable properties: `n_items_to_produce`, representing the number of items still to be produced, and `stopped`, a flag which is set if/when the process needs to be stopped.

An agent program may explicitly define the agent's initial belief-base and the initial task or set of tasks that the agent has to perform, as soon as it is created. In Jason tasks are called *goal* and are represented by Prolog atomic formulae prefixed by an exclamation mark. Referring to the example, the producer agent has an initial task to do, which is represented by the `!produce` goal. Actually, tasks can be assigned also at runtime, by sending to an agent an achieve-goal messages.

```
00  n_items_produced(0).
01  !produce.
02
03  +!produce
04    <- !setup;
05       !produce_items.
06
07  +!setup
08    <- focus("app_board").
09
10  +!produce_items : not n_items_to_produce(0)
11    <- !produce_item(Item);
12       put(Item);
13       -n_items_produced(N);
14       +n_items_produced(N+1);
15       !produce_items.
16
17  +!produce_items : n_items_to_produce(0)
18    <- !finalize.
19
20  +!produce_item(Item) <- ...
21
22  +!finalize : n_items_produced(N)
23    <- println("completed - items produced: ",N).
24
25  -!produce_items
26    <- !finalize.
27
28  +!stopped(true)
29    <- .drop_all_intentions;
30       !finalize.
```

Table I
A PRODUCER AGENT IN JASON.

Then, the main body of an agent program is given by the set of plans, which defines the pro-active and reactive behavior of the agent. The actions contained in a plan body can be split in two categories:

- *internal* actions, that are actions affecting only the internal state of the agent. Examples are actions to create sub-tasks (sub-goals) to be achieved (`!g`), to manage task execution – for instance, to suspend or abort the execution of a task – to update agent inner state – such as adding a new belief (`+b`), removing beliefs (`-b`);
- *external* actions, that are actions provided by the environment, to interact with artifacts. External actions include also communicative actions, which make it possible to communicate with other agents by means of message passing based on speech acts.

Referring to the example, the producer agent has a main plan (line 03-05), which is triggered by an event `+!produce` representing a new goal `!produce` to achieve. Since the agent has an initial `!produce` goal, then this plan will be triggered as soon as the agent is booted. By means of an internal action `!g`, the main plan generates two further subgoals to be achieved sequentially: `!setup` and `!produce_items`.

The plan to handle `!setup` goal (line 07-08) exploits

 ISBN: 978-1-61208-154-0

a predefined action called `focus` to start observing the app_board artifact. Then, two plans are specified for handling the goal `!produce_items`. One (line 10-15) is executed if there are still items to produce—i.e., if the agent has not the belief `n_items_to_produce(0)`. Note that the value of this belief depends on the current state of the app_board resource. This plan first produces a new item (subtask `!produce_item`), then inserts the item in the buffer by means of a `put` action, whose effect is to execute the `put` operation on the resource; if this action succeeds, the plan goes on by updating the belief `n_items_produced` incrementing the number of items produced and generates a new subgoal `!produce_items` to repeat the task. Actually, when executing an external action – such as `put` – it is possible to explicitly denote the artifact providing that action, in order to avoid ambiguities, by means of Jason annotations: `put(Item)[artifact_name("app_board")];`.

The other plan (line 17-18) is executed if there are no more items to produce: in this case the `!finalize` task is executed, which prints on the console the number of items produced by the agent.

The reactive behavior of an agent can be realized by plans triggered by a belief addition/change/removal – corresponding to changes in the state of the environment – and by the failure of a plan in achieving some goal. In the example, the producer agent has a plan (line 28-30) which is executed when the belief `stopped` about the observable property of the artifact is updated to `true`. This means that the user wants to interrupt and stop the production. So the plan stops and drops all the other possible plans in execution – using an internal action `.drop_all_intention` – and the `!finalize` subtask is executed.

Finally, the producer agent has also a plan (line 25-26) to react to the failure of the `!produce_items` task, which is expressed by the event `-!produce_items`. This can happen when the agent, believing that there are still items to be produced, starts the plan to produce a new item and tries to insert it in the buffer. However, the `put` action fails because other agents produced in the meanwhile the missing items.

The semantics of the execution of plans reacting to events is defined by Jason reasoning cycle [10], which is a more articulated version of the execution cycle described in Section III. In particular, the plan stage in this case includes multiple steps, to select – given an event – a plan to be executed. So an agent can have multiple plans in execution but only one action at a time is selected (in the plan stage) and executed (in the act stage). A detailed description of the cycle – as well as of the Jason syntax – can be found in [10].

```
00  public class AppBoard extends Artifact {
01
02    private LinkedList<Object> items;
03    private int bufSize;
04
05    void init(int bufSize, int nItemsToProd){
06      items = new LinkedList<Object>();
07      this.bufSize = bufSize;
08      defineObsProperty("n_item_to_produce",nItemsToProd);
09      defineObsProperty("stopped",false);
10    }
11
12    @OPERATION void put(Object obj){
13      await("bufferNotFull");
14      ArtifactObsProperty stopped =
                                getObsProperty("stopped");
15      if (!stopped.booleanValue()){
16        items.add(obj);
17        ArtifactObsProperty p  =
18             getObsProperty("n_item_to_produce");
19        p.updateValue(p.intValue() - 1);
20      } else {
21       failed("no_more_items_to_produce");
22      }
23    }
24
25    @GUARD boolean bufferNotFull(){
26      return items.size() < nmax;
27    }
28
29    @OPERATION void get(OpFeedbackParam<Object> result){
30      await("itemAvailable");
31      Object item = items.removeFirst();
32      result.set(item);
33    }
34
35    @GUARD boolean itemAvailable(){
36      return items.size() > 0;
37    }
38
39    @OPERATION void stop(){
40      updateObsProperty("stopped",true);
41    }
42  }
```

Table II
THE IMPLEMENTATION OF THE APP_BOARD IN CARTAGO.

C. Programming the Environment in CArtAgO

The implementation of the app_board artifact is shown in Table II. Being CArtAgO a framework on top of the Java platform, artifact-based environments can be implemented using a Java-based API, exploiting the annotation framework. Here we don't go too deeply into the details of such API, we just introduce the main concepts that have been mentioned in Section III; for more information, the interested reader can refer to CArtAgO papers [25] and the documents that are part of CArtAgO distribution [2].

In CArtAgO, an artifact type can be defined by extending a base `Artifact` class. Artifacts are characterized by a usage interface containing a set of operations that agents can execute to get some functionalities. In the example, the artifact app_board provides three operations: `put`, `get` and `stop`. The `put` operation inserts a new element in the buffer – decrementing the number of items to be produced – if the stopped flag has not been set, otherwise the operation (action) fails. The `get` operation removes an item from the buffer, returning it as a feedback of the action. The `stop` operation sets the `stopped` observable property to true.

 ISBN: 978-1-61208-154-0

Operations are implemented by methods annotated with `@OPERATION`. The `init` method is used as constructor of the artifact, getting the initial parameters and setting up the initial artifact state. Inside an operation, guards can be specified (`await` primitive), which suspend the execution of the operation until the specified condition over the artifact state (represented by a boolean method annotated with `@GUARD`) holds. In the example, the `put` operation can be completed only when the buffer is not full (`bufferNotFull` guard) and the `get` one when the buffer is not empty (`bufferNotEmpty` guard). The execution of operations inside an artifact is transactional: among the other things, this implies that at runtime multiple operations can be invoked concurrently on an artifact but only one operation can be in execution at a time–the other ones are suspended. On the agent side, when executing an external action, the agent plan is suspended until the corresponding artifact operation has completed (i.e., the action completed). Then, the action succeeds or fails when (if) the corresponding operation has completed with success or failure. It is worth noting that, in the meanwhile, the agent execution cycle can go on, making it possible for the agent to get percepts and select and perform other actions.

Besides operations, artifacts typically have also a set of observable properties (`n_items_to_produce` and `stopped` in the example), as data items that can be perceived by agents as environment state variables. Instance fields of the class – instead – are used to implement the non observable state of the artifact—for instance, the list of items `items` in the example. Observable properties can be defined, typically during artifact initialization, by means of the `defineObsProperty` primitive, specifying the property name and initial value (line 08-09). Inside operations, observable properties value can be inspected and changed dynamically by means of two basic primitives: `getObsProperty` to retrieve the current value of an observable property (see, for instance, line 14 and 18) and `updateObsProperty` to update the value (line 19).

Besides observable properties, an artifact can make it observable also events occurring when executing operations. This can be done by using a `signal` primitive, specifying the type of the event and a list of actual parameters. For instance, `signal("my_event", "test",0)` generates an observable event `my_event("test",0)`. In the app_board example, to notify the stop we could generate a `stopped` signal in the `stop` operation, instead of using an observable property. Observable events are perceived by all agents observing the artifact—which could react to them as in the case of observable property change.

D. Using JaCa In Real-World Application Contexts

In order to stress the benefits but also the weaknesses of the approach, we are applying this programming model and technology in different application domains.

One is the development of distributed applications based on Service-Oriented Architectures and Web Services in particular. In that context, agents and multi-agent systems are deserving increasing attention both from the applicative viewpoint, as an effective technique to build complex services and applications dynamically composing and orchestrating services [19], and from the foundational viewpoint, as a reference meta-model for the service-based approach, as suggested by the W3C document about Web Services Architecture [3]. To this end, programming models and platforms are needed that make it possible to build SOA/WS applications as agent-oriented systems in a systematic way, exploiting the existing agent languages and platforms to their best, while enabling their co-existence and fruitful cooperation. In that context, we devised a library of artifacts on top of the JaCa platform, enabling the development of SOA/WS applications in terms of workspaces populated by agents and artifacts. Agents encapsulate the responsibility of the execution and control of the business activities and tasks that characterize the SOA-specific scenario, while artifacts encapsulate the business resources and tools needed by agents to operate in the application domain. In particular, artifacts in this case are exploited to model and engineer those parts in the agent world that encapsulate Web Services aspects and functionalities – eventually wrapping existing non-agent-oriented code – to be used, but also changed and adapted by agents at runtime, by need. First results of this work are available here [21].

We are also investigating the approach for the engineering of advanced mobile computing applications, in particular for pervasive and context-aware computing scenarios. To this end, JaCa has been ported on the Android platform [4], enabling the development of Android applications using agent-oriented programming. The project is called JaCa-Android [5]. Actually, besides porting the technology, JaCa-Android includes a library of artifacts that allows agents running into an Android application to seamlessly access and exploit all the features provided by the smartphone and by the Android SDK. Just to have a taste of the approach, Table III shows a snippet of an agent playing the role of smart user assistant, with the task of managing the notifications related to the reception of SMS messages: as soon as an SMS is received, a notification must be shown to the user. A `SMSArtifact` artifact is used to manage SMS messages, in particular this artifact generates an observable event `sms_received` each time a new SMS is received. A `ViewerArtifact` artifact is used to show SMS messages on the screen and to keep track – by means of the `state` observable property – of the current status of the viewer, that is if it is currently visualized by the user on the smartphone screen or not. Finally, a `StatusBarArtifact` artifact is used instead to show messages on the Android status bar, providing a `showNotification` operation to this end. Depending on

 ISBN: 978-1-61208-154-0

```
00  !init.
01
02  +!init
03  <-  focus("SMSArtifact");
04      focus("SMSArtifact");
05      focus("ViewerArtifact").
06
07  +sms_received(Source, Message)
08    : not (state("running") & session(Source)) <-
09    showNotification("jaca.android:drawable/notification",
10       Source, Message, "jaca.android.sms.SmsViewer", Id);
11    +session(Source, Id).
12
13  +sms_received(Source, Message)
14    : state("running") & session(Source)
15    <-  append(Source, Message).
```

Table III
SOURCE CODE OF THE JASON AGENT THAT MANAGES THE SMS NOTIFICATIONS.

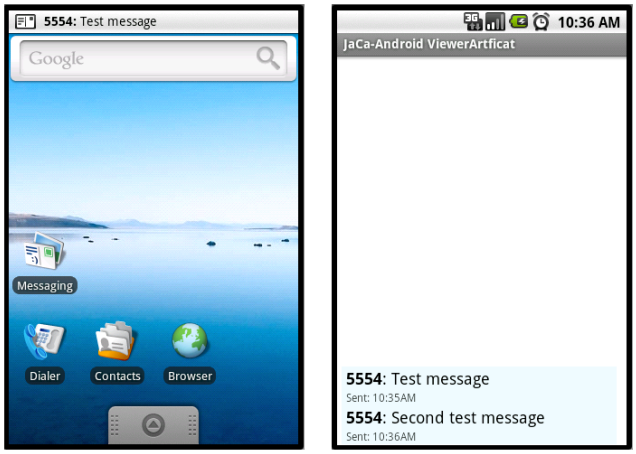


Figure 5. The two different kinds of SMS notifications: (*a*) notification performed using the standard Android status bar, and (*b*) notification performed using the `ViewerArtifact`.

what the user is actually doing and visualizing, the agent shows the notification in different ways. The behavior of the agent, once completed the initialization phase (lines 00-05), is governed by two reactive plans. The first one (lines 7-11) is applicable when a new message arrives and the `ViewerArtifact` is not currently visualized on the smartphone's screen. In this case, the agent performs a `showNotification` action to notify the user of the arrival of a new message using the status bar (Figure 5, *(a)*). The second plan instead (lines 13-15) is applicable when the `ViewerArtifact` is currently displayed on screen and therefore the agent could notify the SMS arrival by simply appending the SMS to the received message list showed by the viewer (Figure 5, *(b)*): this is done by executing the `append` operation provided by `ViewerArtifact`.

From the example, it should be clear that for a developer able to program using the JaCa programming model, moving from one application context to another is a quite straightforward experience. Indeed, she can continue to engineer the business logic of the applications by suitably defining the Jason agent's behavior, and it only need to acquire the ability to work with the artifacts that are specific of the new application context.

E. Current Limitations

On the one side, JaCa allows to exploit in practice some of the benefits of agent-orientation for computer programming described in Section III; on the other side, it suffers of some limitations that we aim at tackling in our future work. Here we consider three main ones.

First, Jason lacks a strong notion of *type*, both for defining the abstract data types used in the programs and for typing agents themselves. This makes agent programs error-prone – some errors are caught only at runtime – and features like inheritance, sub-classing, polymorphism cannot be exploited when developing agents. This is a quite strong limitation due to the fact that such features are the key for providing reusability of the code produced by the developers and therefore are quite essential for: (*i*) the engineering of real-world applications and (*ii*) for the diffusion of the AOP as a mainstream paradigm.

Then, a more seamless integration between the model/platform with the Object-Oriented and Functional programming layer is needed. Currently, for using objects/functions or for integrating any kind of software library (e.g., a library for XML-manipulation), we need to use some sort of wrap mechanism for making them available when programming agents. Now we can realize this sort of wrapping in two ways: (*i*) extending the set of Jason internal actions for directly provide to the agents the required features or (*ii*) encapsulating the required object-oriented/functional-oriented code inside proper artifacts operations.

Finally, Jason plan construct provides a quite weak support for modularizing agent programs. Currently the overall behavior of an agent is defined by a flat list of plans. The absence of a hierarchical structure for plans, explicitly relating plans with sub-plans, could make the understanding of complex agent behavior quite problematic.

V. CONCLUSION

In this paper, we discussed agent-oriented programming as an evolution of Object-Oriented Programming representing the essential nature of decentralized systems where tasks are in charge of autonomous computational entities, which interact and cooperate within a shared environment. We showed in practice some of the main concepts underlying the approach by exploiting the JaCa platform, which is based on existing agent-oriented technologies—the Jason language to program agents and CArtAgO framework to program the environment. However, we believe that, in order to stress and investigate the full value of the agent-oriented approach, a new generation of agent-oriented programming languages is needed, tackling main aspects that have not been considered so far in existing agent technologies –

 ISBN: 978-1-61208-154-0

being not related to AI but to the principles of software development. This is the core of our current and future work, in which we aim at both improving JaCa and eventually exploring the definition of new full-fledged agent-oriented programming languages – so independent from existing technologies – specifically designed since their conception for agent-oriented computing.

REFERENCES

[1] The Foundation of Intelligent Physical Agents organization (FIPA) – http://www.fipa.org, last retrieved: July 5th 2011.

[2] CArtAgO project web site – http://cartago.sourceforge.net, last retrieved: July 5th 2011.

[3] W3C Web Service Architecture – http://www.w3.org/TR/ws-arch/, last retrieved: July 5th 2011.

[4] Android Platform web site – http://developer.android.com, last retrieved: July 5th 2011.

[5] JaCa-Android project web site – http://jaca-android.sourceforge.net/, last retrieved: July 5th 2011.

[6] G. Agha. *Actors: a model of concurrent computation in distributed systems*. MIT Press, Cambridge, MA, USA, 1986.

[7] J. Armstrong. Erlang. *Commun. ACM*, 53:68–75, September 2010.

[8] F. L. Bellifemine, G. Caire, and D. Greenwood. *Developing Multi-Agent Systems with JADE*. Wiley, 2007.

[9] R. Bordini and J. Hübner. BDI agent programming in AgentSpeak using Jason. In F. Toni and P. Torroni, editors, *CLIMA VI*, volume 3900 of *LNAI*, pages 143–164. Springer, Mar. 2006.

[10] R. Bordini, J. Hübner, and M. Wooldridge. *Programming Multi-Agent Systems in AgentSpeak Using Jason*. John Wiley & Sons, Ltd, 2007.

[11] R. H. Bordini, M. Dastani, J. Dix, and A. El Fallah Seghrouchni. *Special Issue: Multi-Agent Programming*, volume 23 (2). Springer Verlag, 2011.

[12] R. H. Bordini, M. Dastani, and A. El Fallah Seghrouchni, editors. *Multi-Agent Programming Languages, Platforms and Applications - Volume 1*, volume 15. Springer, 2005.

[13] R. H. Bordini, M. Dastani, A. El Fallah Seghrouchni, and J. Dix, editors. *Multi-Agent Programming Languages, Platforms and Applications - Volume 2*. Springer, 2009.

[14] P. Haller and M. Odersky. Scala actors: Unifying thread-based and event-based programming. *Theoretical Computer Science*, 2008.

[15] N. Howden, R. Rönnquist, A. Hodgson, and A. Lucas. JACK intelligent agents™ — summary of an agent infrastructure. In *Proceedings of Second International Workshop on Infrastructure for Agents, MAS, and Scalable MAS, held with the Fifth International Conference on Autonomous Agents (Agents 2001)*, 2001.

[16] N. R. Jennings. An agent-based approach for building complex software systems. *Commun. ACM*, 44(4):35–41, 2001.

[17] W. A. Kornfeld and C. Hewitt. The scientific community metaphor, 1981. MIT Artificial Intelligence Laboratory.

[18] H. Lieberman. The continuing quest for abstraction. In *ECOOP 2006*, volume 4067/2006, pages 192–197. Springer, 2006.

[19] M. N. Huhns, M. P. Singh, and M. e. a. Burstein. Research directions for service-oriented multiagent systems. *IEEE Internet Computing*, 9(6):69–70, Nov. 2005.

[20] A. Omicini, A. Ricci, and M. Viroli. Artifacts in the A&A meta-model for multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, 17 (3), Dec. 2008.

[21] M. Piunti, A. Santi, and A. Ricci. Programming SOA/WS systems with BDI agents and artifact-based environments. In *MALLOW-AWESOME*, 2009.

[22] A. Pokahr, L. Braubach, and W. Lamersdorf. Jadex: A BDI reasoning engine. In R. Bordini, M. Dastani, J. Dix, and A. E. F. Seghrouchni, editors, *Multi-Agent Programming*. Kluwer, 2005.

[23] A. S. Rao and M. P. Georgeff. BDI Agents: From Theory to Practice. In *First International Conference on Multi Agent Systems (ICMAS95)*, 1995.

[24] M. Resnick. *Turtles, Termites and Traffic Jams. Explorations in Massively Parallel Microworlds*. MIT Press, 1994.

[25] A. Ricci, M. Piunti, and M. Viroli. Environment programming in multi-agent systems: an artifact-based perspective. *Autonomous Agents and Multi-Agent Systems*, 23:158–192, 2011.

[26] A. Ricci, M. Viroli, and G. Piancastelli. simpA: An agent-oriented approach for programming concurrent applications on top of java. *Science of Computer Programming*, 76(1):37 – 62, 2011.

[27] S. Russell and P. Norvig. *Artificial Intelligence, A Modern Approach (second edition)*. Prentice Hall, 2010.

[28] Y. Shoham. Agent-oriented programming. *Artificial Intelligence*, 60(1):51–92, 1993.

[29] B. C. Smith and C. Hewitt. A plasma primer, 1975. MIT Artificial Intelligence Laboratory.

[30] H. Sutter and J. Larus. Software and the concurrency revolution. *ACM Queue: Tomorrow's Computing Today*, 3(7):54–62, Sept. 2005.

[31] M. D. Travers. *Programming with Agents: New metaphors for thinking about computation*. Massachusetts Institute of Technology, 1996.

[32] M. Wooldridge. *An Introduction to Multi-Agent Systems*. John Wiley & Sons, Ltd, 2002.

 ISBN: 978-1-61208-154-0

Towards Ubiquitous Computing Clouds

Ahmed Khalifa
Bradley Department of Electrical and Computer Engineering, Virginia Tech
Blacksburg, Virginia, USA
Email: akhalifa@vt.edu

Riham Hassan
Computer Science, Virginia Tech
Blacksburg, Virginia, USA
Email: rhabdel@vt.edu

Mohamed Eltoweissy
Bradley Department of Electrical and Computer Engineering, Virginia Tech
Blacksburg, Virginia, USA
Email: toweissy@vt.edu

***Abstract*— With the emergence of ubiquitous resource-infinite computing, the opportunity arises to significantly enhance on-demand resource availability, particularly for under-connected areas, extreme environments and distress situations. To this end, we propose a new concept, termed PlanetCloud, that enables the provisioning of the right, otherwise idle, fixed and mobile resources to effect ubiquitous computing clouds. PlanetCloud utilizes our new opt-in, dynamic spatio-temporal resource calendaring mechanism that aims at providing a dynamic real-time resource scheduling and tracking system. Analysis of dynamic data, from both calendars and social networking, enhances resource forecasting and provides the right-sized cloud resources anytime and anywhere. Our solution enhances computing effectiveness and efficiency by forming local and hybrid clouds to increase cloud availability. In addition, the calendaring mechanism empowers PlanetCloud with an on-demand scalable computing capability through inter-cloud cooperation to provide additional resources beyond single cloud computing capabilities.**

Keywords-Cloud computing; Ubiquitous computing; Social networking; Distributed system; Resource scheduling; Graph theory.

I. INTRODUCTION

The number of mobile nodes is increasing rapidly, which forms massive computing resources. At the same time, there is a need to exploit their idle resources as suggested in [1] Recently, cloud computing has enabled users to exploit outsourced computing power from fixed servers through the Internet [2]. However, Internet connectivity is not always available, especially in rural areas. We advocate the decoupling of computing resources from Internet availability, thus enabling Ubiquitous Computing Cloud (UCC). The UCC utilizes the massive pool of computing resources, in both fixed and mobile nodes.

As a working scenario, consider a resource-provisioning scenario for field missions of the Medicins Sans Frontieres (MSF) organization [3]. MSF provides primary healthcare and assistance to people suffering from distress or even disaster anywhere in the world. Before a field mission is established in a country, a MSF team visits the area to determine the nature of the humanitarian emergency, the level of safety in the area and what type of aid is needed. The field mission might arrive many days after the disaster occurs. A vast quantity of data on several factors, damages and losses, are collected and analyzed during each field mission. However, field missions depend only on their limited resources, which leads to delayed reports to governing bodies to take the right decisions. Moreover, it is difficult to monitor and track the other available resources to cooperate in performing their tasks. Although MSF has consistently attempted to increase media coverage of the situation in these areas to increase international support, there is a lack of support to media coverage due to extreme conditions, e.g. lack of connectivity.

Volunteers of the field missions might travel with their resources among different locations due to the changes in the situation in disaster zones. Some volunteers and infrastructure resources may have withdrawn or be lost due to deteriorated security or disaster damage impact.

MSF, like many other organizations seeking services that require resources, may suffer from the tight coupling between the current cloud computing and the Internet infrastructure. Such coupling sometimes leads to service disruption while decreasing resource availability and computing efficiency. Currently, there is a massive amount of idle computing resources in fixed and mobile nodes. However, we lack an effective resource allocation and scheduling mechanism capable of exploiting these resources to support ubiquitous computing clouds.

A possible solution would be to provide a dynamic real-time resource scheduling, tracking, and forecasting of resources. Further, the solution should enhance the computing efficiency, provide ”on-demand” scalable computing capabilities, increase availability, and enable new economic models for computing service.

We propose a ubiquitous computing clouds’ environment, PlanetCloud, which adopts a novel distributed spatio-temporal calendaring mechanism with real-time synchronization. This mechanism provides dynamic real-time resource scheduling and tracking, which increases cloud availability, by discovering, scheduling and provisioning the right-sized cloud resources anytime and anywhere. Our solution would provide the resources needed by the MSF field missions while they travel among different locations. In addition, it would provide a resource forecasting mechanism by using spatio-temporal calendaring coupled with social network analysis. In situations similar to MSF’s, forecasting of resource availability is invaluable before or even during disaster occurrence to save the time required for both surveys and decision-making. PlanetCloud might discover that uploading or downloading the data to or from a stationary cloud is prohibitively costly in time and money. In this

 ISBN: 978-1-61208-154-0

situation, a group can request resources from PlanetCloud to form an on-demand local clouds or hybrid clouds to enhance the computation efficiency. In case of the MSF scenario, this could be used, if the disaster for example warrants evacuation, to assign evacuation routes for the movement of tens of thousands of evacuees.

PlanetCloud provides "on-demand" scalable computing capabilities by enabling cooperation among clouds to provide extra resources beyond their computing capabilities. In the above scenario, PlanetCloud may provide cooperation among the mission field's cloud and the UN vehicle computing cloud to increase the support of media coverage.

The rest of the paper is organized as follows. In Section II, we give an overview of related works and address the gaps between literature works and our proposed approach. We then detail the architecture in Section III. In Section IV, we present the calendaring mechanism of the proposed approach. We discuss our evaluation approach in Section V. Finally, we present open research issues and conclude this paper in Section VI.

II. RELATED WORKS

The state of the art in research shows four categories of solutions for scheduling and allocating resources relevant to supporting ubiquitous computing clouds. They are resource mapping and discovery techniques, exploiting idle resource approaches, approaches for analysis of complex graphs and networks, and tracking applications.

The resource mapping problem has been broadly considered in the literature. NETEMBED considered allocating resources when deploying a distributed application [4]. Internet Maps are useful for tracking the Internet evolution and studying its properties. In [5], an approach is presented to automatically generate world-wide maps by using traceroute measurement from multiple locations. A detailed survey of various decentralized resource discovery techniques is introduced in [6]. These techniques are driven by the Peer-to-Peer (P2P) network model. A layered architecture was presented to build an Internet-based distributed resource indexing system.

Exploiting idle resources has been proposed in some work. For example, Search for Extra-Terrestrial Intelligence (SETI) @ home focuses on analyzing radio signals, and searching for signs of extra terrestrial intelligence. The software of SETI@home runs either as a screen saver on home computer and only processing data when the screen saver is active, making use of processor idle time [7].

The third category of solutions includes approaches for analysis of complex graphs and networks. Authors in [8] used an extensible data structure for massive graphs: STINGER (Spatio-Temporal Interaction Networks and Graphs (STING) Extensible Representation). It includes a computational approach for the analysis based on the streaming input of spatio-temporal data. GraphCT, a Graph Characterization Toolkit, for massive graphs representing social network data has been presented in [9]. It has been used to analyze public data from Twitter, a micro-blogging network. This work treated social network interactions as a graph and used graph metrics to ascribe importance within the network.

The fourth category includes tracking applications. In [10], an application for the Apple iPhone™ was presented and used to report in real-time flow of traffic in order to build the most accurate map of traffic patterns to be used by commuters or departments of transportation for making decisions. The outcome of the case study is used to determine that the iPhone™ is relatively as accurate as a vehicle tracking device.

Both resource mapping [4, 5, 6] and idle resource exploitation approaches [7] consider stationary resources. Such resources are globally distributed and directly connected to the Internet, as Internet-based systems. On the other hand, PlanetCloud enables ubiquitous computing clouds in a dynamic resource environment by exploiting idle resources that could be either fixed or mobile. In addition, it supports cloud mobility and hybrid cloud formation. PlanetCloud extends the concept of social network analysis presented in [9] to predict and provide the right resources at anytime and anywhere. Moreover, it supports resource infinite computing. An expected limitation of PlanetCloud is providing hard QoS guarantees. PlanetCloud is anticipated to provide soft QoS guarantees based on resource prediction, collection, and stability of environment.

III. OVERVIEW OF PLANETCLOUD

A. Preliminary PlanetCloud Architecture

Our solution provides a novel resource calendaring mechanism deployed on each client. The client calendar is updated dynamically. Initial information about resource availability is collected from clients via server agents in the form of spatio-temporal calendars. User authentication is mandatory before accessing the PlanetCloud system. The user obtains the access credentials from the server agent she registered at before. Calendars and requested tasks are collected from mobile clients, as resource providers, and sent to distributed resource servers. UCCs might be formed to suit the needs of the requesters.

We propose a hierarchical model based on the concept of an agent as a fundamental building block in PlanetCloud. Agents manage tasks proactively without direct user interventions. In addition, agents maintain local data and state functionalities to distributed resource servers (RSs). A RS manages cloud calendars through a data repository storing user profiles, and spatio-temporal calendars of clients and clouds. The PlanetCloud architecture is depicted in Figure 1.

Our proposed spatio-temporal calendaring mechanism enables PlanetCloud to automatically adjust resources to each cloud. Resource allocation abides by the Service Level Agreement (SLA) when conditions or policies change due to user movement [11]. Heterogeneous clients can interact with PlanetCloud throughout the application interface of each

 ISBN: 978-1-61208-154-0

client. There would be multiple interfaces for the calendaring mechanism on different devices.

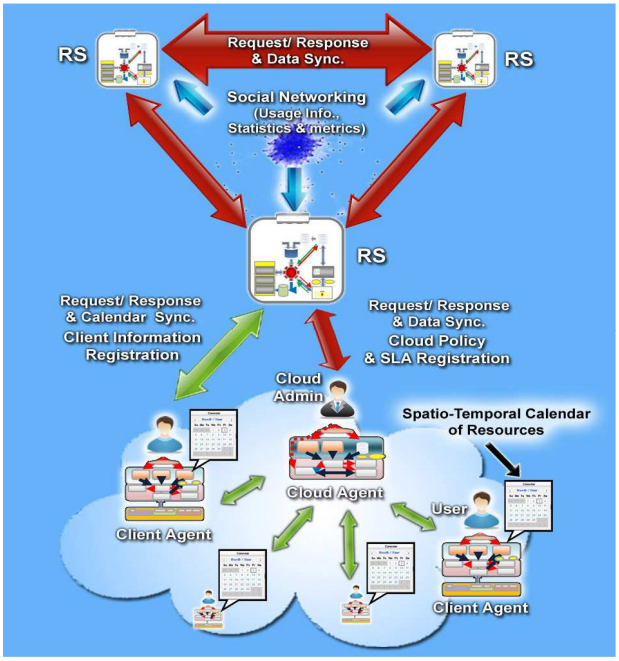


Figure 1. PlanetCloud Architecture.

There are two types of agents in PlanetCloud namely client agents and cloud agents. A single node could be a client agent, a cloud agent, or both.

1) Client Agent

The client agent application manages the client spatio-temporal resource calendar. It handles incoming requests for cloud formation, notifies a user of the next incoming clouds, connects with all other agents involved in the cloud formations, and synchronizes the calendar's content with the RSs data. The client agent consists of two units, the knowledge unit and management unit as shown in Figure 2. The knowledge unit consists of three subunits describing the agent knowledge. The client agent has it own spatio-temporal calendar, which includes spatial and temporal information of the involved resources. The ID-Card unit is used by agents to identify the unique agent ID (AID) as well as the access credentials. The client agent obtains settings about the scheduled/requested clouds and some priority defined parameters through the preferences & settings unit. The Management Unit has five components. The Policy Information Base (PIB) contains predefined or on the fly policies created by a cloud admin, which are used to regulate the operation of PlanetCloud components. The agent manages interactions to form a cloud and prevents inconsistency in the calendar by its controller unit, which provides both policy and client control functions. Agents synchronize their calendar with the data in RSs and cloud agent side through the synchronizer unit. The communicator unit provides the required communications for different activities such as cloud formation requests and responses. Moreover, it is involved in the synchronization process among calendars and PlanetCloud data repositories. The client agent monitors the internal state of the user resources using its observer unit. The lowest layer, of the client agent's building blocks, consists of the application, networking, and computing resources, which are involved in the delivery of the service.

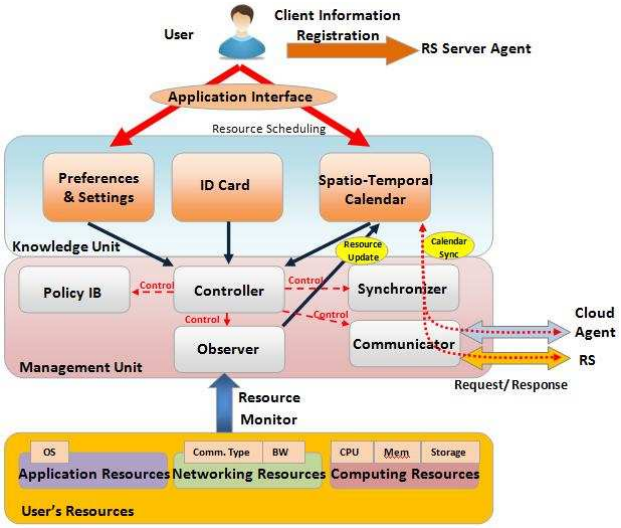


Figure 2. Client Agent Building Blocks

2) Cloud Agent

The cloud agent application is deployed on a high capability client to manage and store the data related to spatio-temporal calendars for all clients within a cloud. Figure 3 shows the building blocks of the cloud agent. The cloud agent uses a central data repository, as a part of its knowledge unit, to store the user profiles, and spatio-temporal calendars of clients within a cloud.

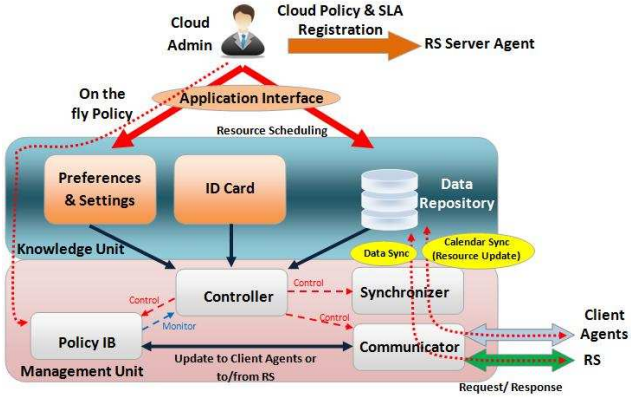


Figure 3. Cloud Agent Building Blocks.

In addition, it includes resource profiles and clients preferences. The cloud agent maintains the overall picture of the resource capability within the cloud. The cloud agent controller can query the data repositories in the RSs for extra resources. Further, the controller provides the necessary resources to the formed cloud based on the SLA information.

 ISBN: 978-1-61208-154-0

3) Resource Server (RS)

Distributed RSs operate on the updated data from clients' calendars and clouds data, which are stored in a data repository as shown in Figure 4. The RS provides resource forecasting using an implemented prediction unit. It uses calendaring data coupled with social network data analysis to increase the forecasting precision. We need to discover the association rules among calendaring data in the data repository. The association rules between different resource types are necessary to define their co-occurrence in each location at different periods of time. We use the calendaring data as following: Generate data files to be excavated; (2) Perform the mining job applying the fuzzy calendar association rules; and (3) Obtain association rules which is extracted from the data files.

We can identify four inputs to the prediction unit namely association rules as an input from calendaring mechanism, and three other inputs from social networks. The inputs from social networks are usage information, usage statistics, and graph type metrics calculated using graph methods. The RS has some other building blocks as follows:

a) Information Bases: It includes two other types, rather than PIB mentioned before namely User Information Base (UIB) and Cloud Information Base (CIB). UIB contains user information (personal, subscribed services, etc.). CIB contains information about the cloud formed by users, SLAs, types of resources needed, amount of each resource type needed, and billing plan for the service.

b) Account manager: It adjusts the bill of a user according to billing policies set by the Cloud Agent.

c) Synchronizer: Repository data are synchronized via updated calendar modifications to/from all clients, updated data modifications to/from all clouds or the other RSs.

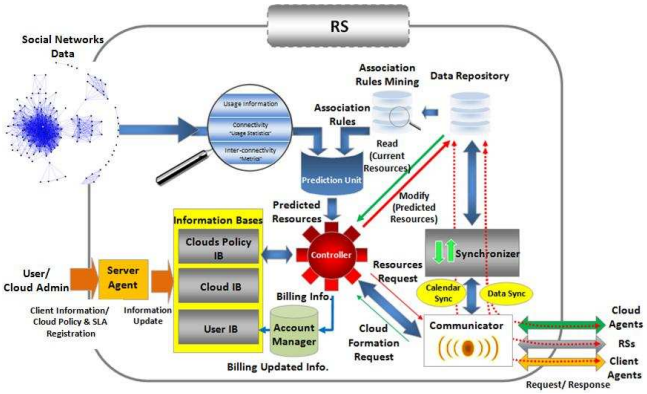


Figure 4. PlanetCloud Resource Server (RS) Building Blocks.

B. Calendar maintenance

We describe our algorithm to maintain the dynamics of the network. The cloud consists of static resources of fixed virtual nodes that meet cloud task requirements. Each virtual node is emulated by a subset of the real mobile nodes, participating in its tasks, while being monitored and tracked by PlanetCloud. Both virtual and mobile nodes have similar communication capabilities allowing them to communicate with one another. The subset locally stores the state of the emulated virtual node. The real nodes perform parts of the task assigned to their emulated virtual node. However, for an emulated virtual node, only one of the real nodes is responsible for aggregating the outputs of the subset, while maintaining their state consistency. If a mobile node fails or leaves the cloud, it ceases to emulate the virtual node; a mobile node that joins the cloud attempts to participate in the emulation.

IV. THE CALENDARING MECHANISM

1) Spatio-temporal resource calendar: We have employed a calendaring schema determined by a hierarchy of calendar units (month, day, hour, minute) [12]. We use a calendar pattern to determine the spatio-temporal resource availability. This pattern includes all time intervals in the corresponding time granularity. A cloud administrator has to specify a calendar pattern which describes his desired periodicities. Then, a query is issued to data repository to locate the proper resources.

Figure 5 shows an example of a spatio-temporal resource calendar. Data could be indexed by three main attributes: time, location, and resources, or combinations of them. Query is performed by any of the main attributes. For example, we can query the availability of certain resource type, or the resources available during a certain time period or at certain location.

Location		Time (Month, Day, Hour, Minute)		Computing resources			Communication resources		Application resources	Cloud ID
X	Y	Start time	End time	Processing	Storage	Memory	Comm. types	Band-width		

Figure 5. Spatio-Temporal Resource Calendar Example.

2) Calendaring Protocol: The cloud administrator requests some tasks through its cloud agent along with the cloud preferences. The cloud agent forwards the cloud formation request to a RS to locate the required resources according to the cloud settings and preferences. The RS looks up the data repository and determines the clients or clouds having the requested resources in terms of time and location domains. The process for scheduling the resources needed for cloud formation is illustrated in Figure 6.

3) Calendaring Application: A software application for calendaring is implemented, where a user can access their spatio-temporal calendars, modify preferences, reschedule his resource availability, and locate resources at anytime and anywhere. In addition, this software enables a user to request an on-demand or hybrid cloud formation,

 ISBN: 978-1-61208-154-0

synchronize the calendar with the PlanetCloud RS data, and be notified about the new requests for his resources.

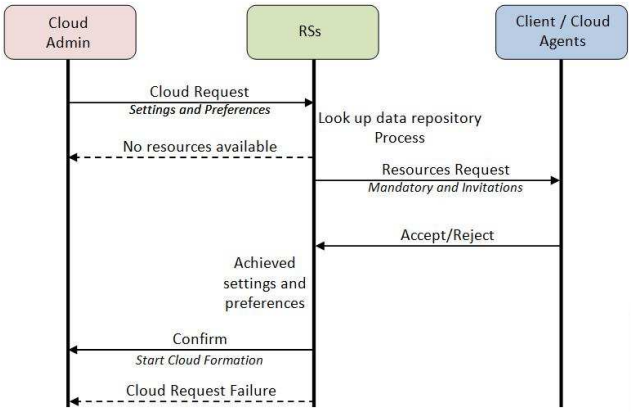


Figure 6. The Cloud Formation Process

V. PERFORMANCE EVALUATION: A DISCUSSION

Our ongoing work focuses on realizing and evaluating the PlanetCloud architecture through a prototype and simulation. The PlanetCloud preliminary prototype spans a specific region by constructing clouds of nodes connected in an ad-hoc manner. We implement an interface to deal with resources of heterogeneous mobile nodes, e.g. mobile devices or laptops. The evaluation will be performed by measuring the query response time, which is the time it takes to answer a query and locate the required resources, and the cloud formation time. In addition, we evaluate the cloud availability in terms of continuity of service in case of failure. We measure the cloud availability by the mean time to recovery & repair from failures. On the other hand, the model for resource forecasting using spatio-temporal calendaring analysis coupled with social network analysis is evaluated by its prediction accuracy, which is measured by the prediction error. Therefore, we measure the difference between the resources correctly predicted and the actual set of resources that are discovered for a given time and location [13]. The parameters for performance evaluation include: number and mobility of nodes as services, calendar size and distribution, and number and type of resources.

VI. CONCLUSION AND FUTURE WORK

In this paper, we proposed the concept of PlanetCloud, which enables ubiquitous computing clouds through using a new dynamic spatio-temporal calendaring mechanism, seamless provisioning of trustworthy resources and socially-intelligent resource forecasting. Our ongoing research comprises the following tasks:

(1) Develop methodology to enhance the analysis performance of complex graphs and clouds with vast quantity of dynamic data from both social networking and spatio-temporal calendars to provide efficient resource discovery, assignment and forecasting. Enhancement is achieved by providing efficient data structure to calculate all association rules among different resource types to find the interesting resource pattern.

(2) Transparently maintain applications' QoS by providing an efficient mechanism to maintain local data and state functionalities in a highly dynamic environment. This mechanism provides virtual static resources that meet cloud task requirements, while being emulated by subsets of the real mobile nodes, which maintain their state consistency.

(3) Provide a solution to achieve privacy preservation when sharing resources and data for calendaring. There is a need to support distributed data access and computations without compromising the raw data of cloud nodes.

REFERENCES

[1] M. Eltoweissy, S. Olariu, and M. Younis, "Towards autonomous vehicular clouds," in Ad Hoc Networks, ser. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering. Springer Berlin Heidelberg, 2010, vol. 49, pp. 1–16.

[2] C. Wang, K. Ren, W. Lou, and J. Li, "Toward publicly auditable secure cloud data storage services," IEEE Network, vol. 24, no. 4, pp. 19 –24, 2010.

[3] B. Kouchner, "Globalization and new world order: are we ready for "scientists without borders"?" IEEE Transactions on Applied Superconductivity, vol. 15, pp. 1071 – 1077, 2005.

[4] J. Londono and A. Bestavros, "Netembed: A network resource mapping service for distributed applications," IEEE International Symposium on In Parallel and Distributed Processing, 2008. IPDPS 2008., 2008, pp. 1 –8.

[5] Y. Shavitt and N. Zilberman, "A structural approach for pop geo-location," in INFOCOM IEEE Conference on Computer Communications Workshops , 2010, 2010, pp. 1 –6.

[6] R. Ranjan, A. Harwood, and R. Buyya, "Peer-to-peer-based resource discovery in global grids: a tutorial," IEEE Communications Surveys Tutorials, vol. 10, no. 2, pp. 6 –33, 2008.

[7] D. Werthimer, J. Cobb, M. Lebofsky, D. Anderson, and E. Korpela, "Seti@homemassively distributed computing for seti," Computing in Science and Eng., vol. 3, pp. 78–83, January 2001.

[8] D. Ediger, K. Jiang, J. Riedy, and D. Bader, "Massive streaming data analytics: A case study with clustering coefficients," 2010 IEEE International Symposium on In Parallel Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010, pp. 1 –8.

[9] D. Ediger, K. Jiang, J. Riedy, D. Bader, C. Corley, R. Farber, and W. Reynolds, "Massive social network analysis: Mining twitter for social good," 39th International Conference on In Parallel Processing (ICPP), 2010, pp. 583 –593.

[10] T. Menard, and J. Miller, "FreeSim_Mobile: A novel approach to real-time traffic gathering using the apple iPhone," IEEE in Vehicular Networking Conference (VNC), 2010, pp. 57 -63, Dec. 2010.

[11] Y. Cheng, R. Farha, M. Kim, A. Leon-Garcia, and J. W.- K. Hong, "A generic architecture for autonomic service and network management," Computer Communications, no. 29, pp. 3691–3709, 2006.

[12] J.-Y. Jiang, W.-J. Lee, and S.-J. Lee, "Mining calendar-based asynchronous periodical association rules with fuzzy calendar constraints," in 14th IEEE International Conference on Fuzzy Systems, 2005, pp. 773–778.

[13] S. Lili, Y. Shoubao, G. Liangmin, and W. Bin, "A Markov chain based resource prediction in computational grid," in Fourth International Conference on Frontier of Computer Science and Technology (FCST), pp. 119 –124.

 ISBN: 978-1-61208-154-0

The Hopfield-type Memory Without Catastrophic Forgetting

Iakov Karandashev Boris Kryzhanovsky Leonid Litinskii
Yakov.Karandashev@phystech.edu kryzhanov@mail.ru litin@mail.ru
Center of Optical Neural Technologies of Scientific Research Institute for System Analysis
Russian Academy of Science
Moscow, Russia

***Abstract*—We analyzed a Hopfield-like model of artificial memory that reproduces some features of the human memory. They are: a) the ability to absorb new information when working; b) the memorized patterns are only a small part of a set of patterns that are written down in connection matrix; c) the more the pattern was shown during the learning process, the better the quality of its recognition. We used the Hebb rule, but each pattern was supplied with its own weight. The weight is proportional to the frequency of the pattern showing during the learning of the network. As a result unlimited number of patterns can be written down in the connection matrix, and that would not lead to the memory destroying (as it has place in the standard Hopfield model). However, only the patterns that were shown rather frequently would be recognized: their weights have to be larger a critical value. For analyzed variants of the weights distribution the storage capacity was estimated as ~0.05*N*-0.06*N*, where *N* is the dimensionality of the problem.**

Keywords - Associative memory; catastrophic forgetting; quasi-Hebbian matrix.

I. INTRODUCTION

In the standard Hopfield model one uses the Hebb matrix constructed with the aid of M random patterns [1], [2]. For definiteness we suppose that *N*-dimensional vector-row $\mathbf{x}^\mu = (x_1^\mu, x_2^\mu, ..., x_N^\mu)$ with binary coordinates $x_i^\mu = \pm 1$ is the μ-th pattern, the number of patterns is equal to M, and the Hebb connection matrix has the form

$$J_{ij} = (1-\delta_{ij})\sum_{\mu=1}^{M} x_i^\mu x_j^\mu \ .$$

The estimate for number of random patterns that can be memorized in the Hopfield model is well known: $M_c \approx 0.138 \cdot N$. If the number of patterns written down into connection matrix is larger than M_c, the catastrophe takes place: the network ceases to recognize patterns at all.

This catastrophic forgetting is a troublesome defect of the Hopfield model. Indeed, let us imagine a robot whose memory is based on the Hopfield model. It is natural to think that his memory is steadily filled up. When the robot sees a new image, it is written additionally to its memory. Catastrophic forgetting means that when the number of memorized patterns exceeds M_c, the memory will be completely destroyed. Everything that was accumulated in the memory would be forgotten. This behavior is contrary to common sense.

Earlier some modifications of the Hebb matrix were proposed to eliminate the memory destruction [3]-[7]. As the result of such modifications an unlimited number of random patterns can be fearlessly written down into matrix elements one by one. However, the memory of the network is restricted. If as previously the maximum number of recognized patterns is denoted by M_c, for the models discussed in [3]-[7] $M_c \approx 0.05 \cdot N$. All these models have the same weak point: only patterns that are the last written down in the connection matrix constitute the memory of the network. Let us explain the last statement. Let us number the patterns in the course of their appearance during the learning process: the later the pattern was shown to the network, the larger its number. Then it turns out that the real network memory is formed of patterns whose numbers belong to the interval $\mu \in [M - M_c, M]$. Patterns with order numbers less than $M - M_c$ are excluded from the memory irretrievably. That is the common property of the models [3]-[7].

In our work, we succeeded in eliminating of the catastrophic forgetting of the Hopfield model. In our approach every pattern is supplied by an individual weight. Then in place of the Hebbian matrix we obtain a quasi-Hebbian connection of the form

$$J_{ij} = (1-\delta_{ij})\sum_{\mu=1}^{M} r_\mu x_i^\mu x_j^\mu \ . \qquad (1)$$

Here the weights r_μ are positive and put in decreasing order: $r_1 \geq r_2 \geq ... \geq ... \geq 0$. It is natural to treat the weight r_μ as the frequency of the μ-th pattern showing during the learning process.

In previous papers [8], [9] with the aid of statistical physics methods the main equation for the Hopfield model with the quasi-Hebbian matrix was obtained. This equation generalizes the classical equation of the standard Hopfield model [1]. For a special distribution of the weights we succeeded in solution of the main equation: the case when only one coefficient differed from all other that were identically equal: $r_1 = \tau$, $r_2 = r_3 = ... = r_M = 1$ was examined. Theoretical results were confirmed by computer simulations. (In what follows we use the notations introduced in [8], [9].)

In this paper, we present the results of solving of the main equation in the general case. The main obtained result is as follows. For every weights distribution $\{r_\mu\}$ there is such a critical value r_c that only patterns whose weights are greater than r_c will be recognized by the network. Other

 ISBN: 978-1-61208-154-0

patterns are not recognized. Now we know only the algorithm of calculation of the critical value r_c. It is not clear, if it is possible to obtain an analytical expression for r_c.

The case of the weights, which are the terms of a decreasing geometric series $r_\mu = q^\mu$ ($q<1$) we discuss in details. For this weights distribution the number of the recognized patterns is $\sim 0.05 \cdot N$. The results are confirmed by computer simulations.

Note, for the first time the quasi-Hebbian connection matrix (1) was discussed many years ago. For this matrix the implicit form of the main equation (2) was obtained in [6]. However, the authors of [6] examined the case of the standard Hopfield model only ($r_\mu \equiv 1$). Our contribution is the solution of the main equation in the general form.

II. SOLUTION OF MAIN EQUATION

The main equation for the k-th pattern, which we obtained in [8], [9], has the form:

$$\frac{\gamma^2}{\alpha} = \frac{1}{M-1}\sum_{\mu\neq k}^{M}\left(\frac{r_\mu}{r_k\varphi - r_\mu}\right)^2 . \qquad (2)$$

It is supposed that M and N are very large: $M, N >> 1$, α is the load parameter: $\alpha = M/N$, $\gamma = \gamma(y)$ and $\varphi = \varphi(y)$ are functions of an auxiliary argument $y \geq 0$:

$$\gamma(y) = \sqrt{\frac{2}{\pi}}e^{-y^2}, \quad \varphi(y) = \frac{\sqrt{\pi}erf(y)}{2y}e^{y^2}.$$

The function $\gamma(y)$ decreases monotonically, and $\varphi(y)$ increases monotonically from the minimal value $\varphi(0)=1$. For what follows it is important that $\varphi(y) \geq 1$. When all the weights are equal to each other, equation (2) turns into the equation for the standard Hopfield model [1], [2], [6].

Let us transform Eq.(2) dividing the left hand and right hand sides by M. Moreover we tend the upper limit of the sum in r.h.s. of Eq. (2) to infinity. In fact, we pass to the model with infinitely number of patterns. The main equation takes form:

$$N = \sum_{\mu\neq k}^{\infty} f_\mu^{(k)}(y), \qquad (3)$$

where $f_\mu^{(k)}$ are the functions of γ, φ and r_μ:

$$f_\mu^{(k)}(y) = \left(\frac{t_\mu^{(k)}}{\gamma(\varphi - t_\mu^{(k)})}\right)^2, \; t_\mu^{(k)} = \frac{r_\mu}{r_k}, \; \mu \neq k . \qquad (4)$$

Eq. (3) connects the dimensionality N of the problem with number k of the pattern. When we find the solution y_0 of this equation, we calculate the overlap of the k-th pattern with the nearest fixed point:

$$m_k = erf(y_0). \qquad (5)$$

When $m_k \approx 1$, in the vicinity of the k-th pattern there is a fixed point of the network. This situation is interpreted as recognition of the k-th pattern by the network. If $m_k \approx 0$, the k-th pattern is not recognized by the network.

The values $t_\mu^{(k)}$ are arranged in decreasing order. For what follows it is important that the first $k-1$ of these values are larger than 1, and the other ones are less than 1:

$$t_1^{(k)} > t_2^{(k)} > ... > t_{k-1}^{(k)} > 1 > t_{k+1}^{(k)} > t_{k+2}^{(k)} > .... \qquad (6)$$

The r.h.s. of Eq. (3) is the result of summarizing over the set of functions $f_\mu^{(k)}(y)$ (4). It is easy to see that when $y \to \infty$ the denominator $\gamma(\varphi - t_\mu^{(k)})$ of any function $f_\mu^{(k)}(y)$ tends to 0. In other words, at the infinity each function $f_\mu^{(k)}(y)$ increases unrestrictedly. As a result, at the infinity the r.h.s. of Eq.(3) increases unrestrictedly too.

The behavior of the function $f_\mu^{(k)}(y)$ for finite values of argument depends on the constant $t_\mu^{(k)}$ in its denominator. If $t_\mu^{(k)} < 1$, the function $f_\mu^{(k)}(y)$ is everywhere continuous and limited. If $t_\mu^{(k)} > 1$, the function $f_\mu^{(k)}(y)$ has a singular point. In this case the denominator of the function $f_\mu^{(k)}(y)$ is equal to zero for some value $y_\mu^{(k)}$ of its argument:

$$\varphi\left(y_\mu^{(k)}\right) = t_\mu^{(k)} \quad \Leftrightarrow \quad y_\mu^{(k)} = \varphi^{-1}\left(t_\mu^{(k)}\right),$$

where φ^{-1} is the inverse function with regard to φ. We see that for every $t_\mu^{(k)} > 1$ the function $f_\mu^{(k)}(y)$ has the discontinuity of the second kind in the point $y_\mu^{(k)}$. Since in the series (6) the first $k-1$ values of $t_\mu^{(k)}$ are greater than 1, it is easy to understand that the r.h.s. of Eq.(3) has the discontinuities of the second kind in $k-1$ points $y_{k-1}^{(k)} < y_{k-2}^{(k)} < ... < y_1^{(k)}$. At the infinity the r.h.s. of Eq.(3) increases unrestrictedly.

For simplicity let us go in Eq.(3) to inverse quantities:

$$\frac{1}{N} = F_k(y), \text{ where } F_k(y) = \left(\sum_{\mu\neq k}^{\infty} f_\mu^{(k)}(y)\right)^{-1}. \qquad (7)$$

It is evident that nonnegative function $F_k(y)$ in the r.h.s. of Eq.(7) is equal to zero in the points $y_{k-1}^{(k)} < y_{k-2}^{(k)} < \ldots < y_1^{(k)}$. At the infinity $F_k(y)$ tends to zero. The typical behavior of the function $F_k(y)$ is shown in Fig.1. To the right of the rightmost zero $y_1^{(k)}$, where the inequality $\varphi(y) > t_1^{(k)}$ holds, the function $F_k(y)$ at first increases, and then after reaching its maximum the function $F_k(y)$ decreases monotonically. Let $y_c^{(k)}$ be the coordinate of the rightmost maximum of the function $F_k(y)$. The value of $F_k(y)$ in the point $y_c^{(k)}$ determines the critical characteristics related to the recognition of the k-th pattern. Let us explain what it means.

 ISBN: 978-1-61208-154-0

Generally speaking, Eq.(7) has several solutions. Their number is equal to the number of intersections of the function $F_k(y)$ with the straight line that is parallel to abscissa axis at the height $1/N$ (see Fig. 1). These solutions correspond to stationary points of the solution of the saddle-point equation [1], [2]. However, only one of these intersections is important. Its coordinate is to the right of the rightmost maximum $y_c^{(k)}$. This intersection corresponds to the minimum of the solution of the saddle-point equation. Other solutions of Eq.(7) can be omitted.

As example in Fig.1 the behavior of the r.h.s. of Eq.(7) for the pattern number 5 ($k=5$) is shown for the weights that are the terms of harmonic series $r_\mu = 1/\mu$. Four points $y_4^{(5)} < y_3^{(5)} < y_2^{(5)} < y_1^{(5)}$ are zeros of the function $F_5(y)$. For y that are greater than $y_1^{(5)}$ ($y > y_1^{(5)}$) the function $F_5(y)$ at first increases up to its local maximum in the point $y_c^{(5)}$ and then decreases monotonically. The dashed line that is parallel to the abscissa axis is drawn at the height 0.001. When the l.h.s. of Eq.(7) is equal to 0.001, we obtain $N = 1000$. In other words, for this quasi-Hebbian matrix of the dimensionality $N = 1000$ in the vicinity of the 5-th pattern there is a fixed point certainly. Since the solution of Eq.(7) is large enough, $y_0 \approx 3.5$, the overlap (5) of the pattern and the fixed point is close to 1.

Let us little by little decrease the dimensionality N. The dashed straight line will go up, and the solution $y_0(N)$ of Eq.(7) will shift in the region of smaller values. This will go on till y_0 coincides with the critical value $y_c^{(5)}$. Just this defines the minimal dimensionality $N_{\min}$ for which the fixed point in the vicinity of the 5-th pattern still exists. Since for $N < N_{\min}$ equation (7) has no solutions in the region $y > y_1^{(5)}$, there is no fixed points in the vicinity of the 5-th pattern. From the point of view of the neural network memory this means that when $N < N_{\min}$ the 5-th pattern is not recognized by the network.

Up to now we fixed the number of the pattern k and decreased the dimensionality N. However, it is reasonable to fix the dimensionality N and increase k little by little. We seek its maximal value for which Eq.(7) has a solution. It is easy to show that when k increases the critical point $y_c^{(k)}$ shifts to the right, and the value of the maximum of $F_k(y_c^{(k)})$ decreases steadily. Then it is not difficult to find the maximal value of k for which Eq.(7) has a solution. By $k_m = k_m(N)$ we denote this maximal value.

For given dimensionality N the pattern with the number k_m is the last in whose vicinity there is a fixed point. For $k < k_m$ Eq.(7) has a solution in the region $y > y_c^{(k)}$ too. Consequently, these patterns will also be recognized. On the contrary, for $k > k_m$ Eq.(7) has no solutions in the region $y > y_c^{(k)}$. Consequently, the patterns with such numbers will not be recognized.

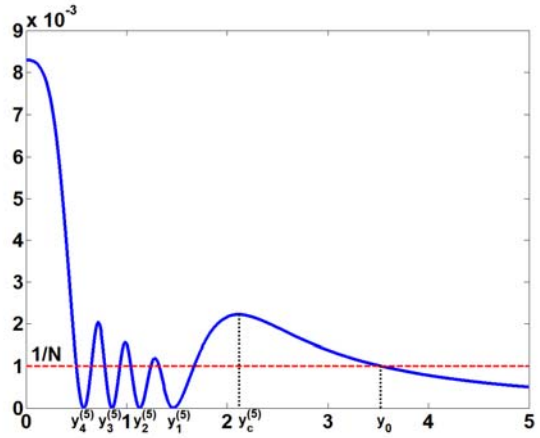


Figure 1. The behavior of the function $F_k(y)$ defined by Eq.(7) when the weights are equal to $r_\mu = 1/\mu$. Here $k=5$, y_0 is the solution of the equation (7) for $1/N = 0.001$ and $y_c^{(5)}$ is the critical value.

Let r_c be the weight corresponding to the pattern with the number k_m: $r_c = r_{k_m}$. Our consideration shows that only the patterns, whose weights are not less than the critical value r_c will be recognized. Patterns whose weights are less than the critical value r_c are not recognized in spite of the fact that these patterns take part in the construction of the quasi-Hebbian matrix. The memory of the network is limited, but the catastrophic forgetting does not occur.

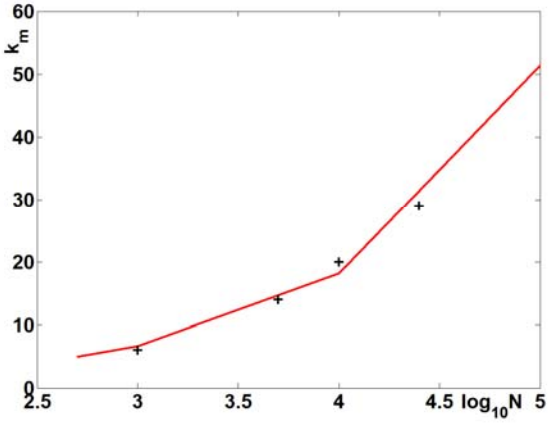


Figure 2. Maximal number of the pattern $k_m(N)$ that can be recognized for the weights $r_\mu = 1/\mu$. The daggers show the results of computer simulations $k_m^{\exp}$.

Let us show how this approach works, choosing as an example matrices whose weights are the terms of harmonic

series $r_\mu = 1/\mu$. For the dimensionalities $N = 10^3$, $5\cdot10^3$, 10^4, $2.5\cdot10^4$ and 10^5 we obtained the maximal number $k_m(N)$ of the pattern that could be recognized solving Eq.(7) numerically. In Fig.2 the value of $k_m(N)$ as function of $\log_{10} N$ is shown. The daggers are the results of computer simulation $k_m^{\exp}$. For the given N we generated a random matrix with the weights that are the terms of harmonic series and defined the maximal number of the pattern that was a fixed point. (Patterns with larger numbers differed notably from the nearest fixed points.) The number $k_m^{\exp}$ was obtained as a result of averaging over 10 random matrices. We see that the experimental values are in good agreement with the theoretical results. The numerical solution of Eq.(7) shows that the storage capacity of such a network asymptotically tends to zero:

$$\lim_{N\to\infty} k_m(N)/N \sim \lim_{N\to\infty} 1/\sqrt{N\ln N} = 0 .$$

In the next section we examine in details a particular distribution of the weights.

III. GEOMETRIC PROGRESSION AS WEIGHTS

Let us discuss in details the case of the weights in the form of decreasing geometric progression $r_\mu = q^\mu$, where $q \in (0,1)$. For the first the weight of such a type were mentioned in [5] and [6]. It is natural to assume that in Eq.(3) the first value of the summation index is equal to zero and $r_0 = 1$. Now Eq.(3) has the form

$$N = \frac{1}{\gamma^2}\sum_{\mu=0\neq k}^{\infty}\left(\frac{q^\mu}{s_k - q^\mu}\right)^2, \text{ where } s_k = q^k\varphi(y). \qquad (8)$$

Our interest is the solution of this equation for large values of the argument when the inequality $s_k = q^k\varphi(y) > 1$ is fulfilled. In the r.h.s. of Eq.(8) we replace summation by integration. Then in both sides of the equation we pass to inverse quantities and obtain the analogue of equation (7):

$$\frac{1}{N} = \frac{\gamma^2(\varphi-1)^2}{(\varphi-1)^2\Phi_k(y) - 1}, \qquad (9)$$

where

$$\Phi_k(y) = \frac{\ln\left(\dfrac{s_k - 1}{s_k}\right) + \dfrac{1}{s_k - 1}}{|\ln q|} .$$

When solving Eq.(9) numerically we can find the maximal number of the pattern k_m which is recognized by the network yet, $k_m = k_m(N,q)$. Our goal is to find such value of the parameter q that corresponds to the maximum of the storage capacity of the network. In other words, we find the value of the parameter q for which $k_m(N,q)$ is maximal: $\tilde{k}(N) = \max_q k_m(N,q)$. It is evident that this optimal q have to exist; $q_m = q_m(N)$ denotes its value.

On the left panel of Fig.3 the dependence of the ratio $k_m(N,q)/N$ on q for three dimensionalities N is shown. We see that curves have distinct points of maximum, but for all cases the value of the all maximums are the same:

$$\lim_{N\to\infty} \tilde{k}(N)/N \approx 0.05 . \qquad (10)$$

In other words, the maximal number of patterns that can be memorized by the network is expressed as $M_c \approx 0.05\cdot N$. It is more than two times less than the storage capacity for the standard Hopfield model ($0.138\cdot N$), but the catastrophic destruction of the memory does not occur. Let us note that the optimal values $q_m(N)$ are rather close to 1: $q_m = 0.992, 0.9992$ and 0.99992 for N =1000, 10000 and 100000, respectively.

When the value of q becomes larger than q_m, the number of recognized patterns decreases. It is clear that as far as the parameter q tends to 1, our model more and more resemble the standard Hopfield model for which the dimensionality N is finite and the number of the patterns M is infinitely large. It is clear that when $q = 1$ the network memory is destroyed: patterns are not recognized by the network. It turns out that the destruction of the memory occurs even before q becomes equal to 1.

Let q_c denotes the critical value of q under which only the first pattern (with the maximal weight) is recognized; it is clear, that $q_c > q_m$. It is possible to obtain an analytic estimate for q_c: $q_c = 1-\delta$, where $\delta \approx 1/0.329N$. Up to now we do not succeeded in analytic estimation of q_m. From the fitting of the results of the numerical solution of Eq.(9) we obtain: $q_m \approx 1 - 2.75\cdot\delta$.

For the last pattern with the number k_m that can be recognized by the network, on the right panel of Fig.3 the dependence of the overlap on q is shown. In the point of the solution "breakdown" q_c all overlaps have approximately the same values $m_c \approx 0.933$.

The results of this section were obtained by means of numerical solution of eq. (9). At the present time we try to obtain these results analytically. Moreover, it is necessary to compare our predictions with the computer simulations in the case when the weights are terms of geometrical progression. First experiments show good agreement with theoretical predictions. It is interesting to analyze other distributions of the weights r_μ differ from a decreasing geometrical progression. Now these investigations are in progress.

IV. CONCLUSIONS

The common experience indicates that the learning process is a continuous one. In the brain there are mechanisms allowing one to accumulate steadily the obtained information. The Hopfield model is an artificial model of the human ability of learning. For this reason the

catastrophic destruction of the memory due to too much number of patterns is absolutely inadmissible. An artificial memory of a robot also has to work even if it obtains new information continuously written down.

The method of eliminating of catastrophic destruction of the memory, proposed in our paper, seems to be very attractive. It turns out that it is possible to learn the network uninterruptedly, even during the process of the patterns recognition. Indeed, each pattern that finds itself in the field of vision of a robot can be automatically added to the connection matrix. Each pattern modifies matrix elements according to the standard Hebbian rule. If the pattern is the same as the one written down previously, its weight increases by 1. If the pattern is new, it is written down into the connection matrix with the initial weight equals to 1. According to the statistics of the patterns appearance the weights distribution is online modified. There is no catastrophic destruction of the memory since every moment the real memory of the network consists of patterns, whose weights are larger than the current critical value r_c .

Let the weight of a pattern be less than r_c but we need this pattern to be recognized by the network. It is sufficient to increase the weight of this pattern doing it to be larger than the critical value, and this pattern will be recognized. It is possible that at the same time some other patterns cease to be recognized. (They are those whose weights are only slightly exceeds the critical value r_c). Such replacement of patterns by other ones does not contradict to the common sense. It corresponds to the general conception of the human memory.

ACKNOWLEDGMENT

The work was supported by the program of Russian Academy of Sciences “Information technologies and analysis of complex systems” (project 1.7) and in part by Russian Basic Research Foundation (grant 09-07-00159).

REFERENCES

[1] D. Amit, H. Gutfreund, and H. Sompolinsky, “Statistical mechanics of neural networks near saturation,” Annals of Physics, vol. 173, pp. 30-67, 1987.

[2] J. Hertz, A. Krogh, and R. Palmer, Introduction to the Theory of Neural Computation. Massachusetts: Addison-Wesley, 1991.

[3] G. Parisi, “A memory which forgets,” Journal of Physics A vol. 19, pp. L617-L620, 1986.

[4] J.P. Nadal, G. Toulouse, J.P. Changeux, and S. Dehaene, “Networks of Formal Neurons and Memory Palimpsest,” Europhysics Letters vol.1(10), pp. 535-542, 1986.

[5] J.L. van Hemmen, G. Keller, and R. Kuhn, “Forgetful Memories,” Europhysics Letters, vol. 5(7), pp. 663-668, 1988.

[6] J.L. van Hemmen and R. Kuhn, “Collective Phenomena in Neural Networks,”. in Models of Neural Networks, E. Domany, J.L van Hemmen and K. Shulten, Eds. Berlin: Springer, 1992, pp. 1-105.

[7] A. Sandberg, O. Ekeberg, and A. Lansner, “An incremental Bayesian learning rule for palimpsest memory,” preprint.

[8] Ja. Karandashev, B. Kryzhanovsky, and L. Litinskii, “Local Minima of a Quadratic Binary Functional with a Quasi-Hebbian Connection Matrix,” Proc. 20th International Conference on Artificial Neural Networks (ICANN’10) Springer, 2010, Part 3, pp. 41-50, LNCS 6352.

[9] Ya. Karandashev, B. Kryzhanovsky, and L. Litinskii, “Local Minima of a Quadratic Binary Functional with a Quasi-Hebbian Connection Matrix,” arXiv:1012.4981v1.

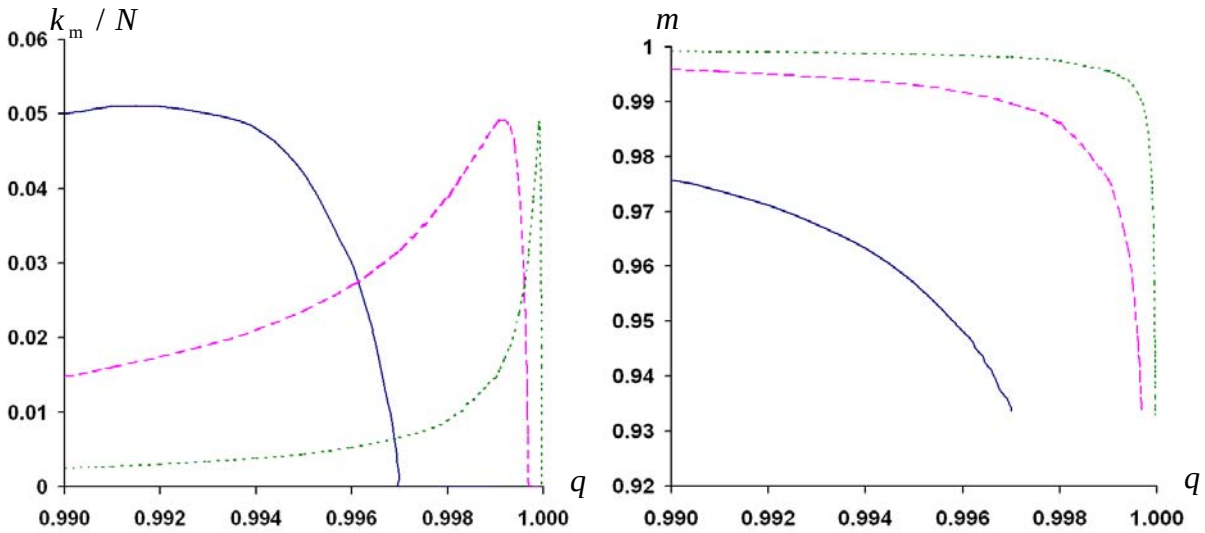


Figure 3. For three dimensionalities N we show: on the left panel the dependence k_m / N on q; on the right panel synchronous values of the pattern overlaps with the nearest fixed point. On both panels the solid line corresponds to the dimensionality N=1000, the dashed line corresponds to N=10000, the point line corresponds to N=100000.

 ISBN: 978-1-61208-154-0

An Artificial Intelligence Approach Towards Sensorial Materials

Florian Pantke, Stefan Bosse, Michael Lawo, Dirk Lehmhus
ISIS Sensorial Materials Scientific Center
University of Bremen
Bibliothekstrasse 1, 28359 Bremen, Germany
{*fpantke, sbosse, mlawo*}*@informatik.uni-bremen.de,*
dirk.lehmhus@uni-bremen.de

Matthias Busse

Fraunhofer IFAM
Wiener Strasse 12, 28359 Bremen, Germany
matthias.busse@ifam.fraunhofer.de

***Abstract*—Sensorization aims at equipping technical structures with an analog of a nervous system by providing a network of sensors and communication facilities that link them. The objective is that, instead of having been designed to loads and tested to conditions, a structure can experience and report design constraint violations by means of real-time self-monitoring. Specialized electronic components and computational algorithms are needed to derive meaning from the combined signals of integrated sensors. For this task, artificial intelligence approaches constantly gain importance; the more so as the trend of ever increasing sensor network size and density suggests that sensor and structure may soon become one, forming a sensorial material. Current simulation techniques capture many aspects of sensor networks and structures. For decision making and communication, the intelligent agent paradigm is an accepted approach, as is finite element analysis for structural behavior. To gain knowledge how sensorial structures can most effectively be built, an artificial intelligence based process for the design of such structures was developed that uses machine learning methods for fast load inference. It is presented in this paper, along with evaluation results obtained in experiments using a finite element model of a strain gauge equipped plate, which demonstrate the general practicability.**

***Keywords*-sensorial material; finite element method; sensor network; machine learning; multi-agent system**

I. INTRODUCTION

Adopting principles from nature, sensorization aims at equipping technical structures with an analog of a nervous system by providing a network of sensors and communication facilities that link them. Specialized electronic components as well as suitable computational algorithms are needed to derive meaningful information from the sensor signals. The main objective is to build structures that—instead of being designed once and tested for health based on externally observable exceptional events or in predefined intervals—constantly monitor themselves during operation by means of sensors attached to the structure's surface, directly printed on it [1], or embedded inside the material [2]. Such structures will be able to infer facts about their current loading state, e.g., autonomously detect overloading, as well as permanently record and carry with them their entire mechanical and/or thermal loading history. In order to attain high sensor densities, i.e., to embed hundreds or thousands of sensors into one single structure, all sensors and electronic circuits need to be miniaturized to a maximum degree. As these components usually have no significant load-bearing capabilities, the volume they consume within the structure directly affects the latter's mechanical stability. Thus, in the sensor integration process, avoiding the introduction of evident macrostructural defects is imperative [3] [4].

An additional requirement, particularly in mobile applications, is a low power consumption profile of the electronic components. It simply is not feasible in most cases to equip the structure with a high-performance microprocessor unit along with sufficient RAM that would be necessary for inferring facts about the applied load cases from the sensor signals by employing the finite element method (FEM) during operation, for instance. Less computation- and resource-intensive approaches are called for. Especially if smart structures reacting to stimuli from their environment [5] are desired, a spatial distribution of numerous simple, miniaturized, low-power evaluation units over the entire structure seems advisable. As is the case for a human accidentally touching a hot plate, fast locally confined reactions preventing (further) damage may be vital for an engineered structure, while calculating, e.g., the millimeter-precise location of load introduction along with the global geometric state of the entire structure may be an unaffordable luxury. For long-term availability, the respective sensor nodes should ideally draw on local energy sources [6].

This article presents a conceptual approach to designing sensorial structures and implementing their self-monitoring capabilities utilizing basic methods from the field of artificial intelligence (AI). The central aim is to derive useful information despite having to cope with limited computational power and with noise in sensor signals. Lack of calibration as well as of matching between sensors and the possibility of drift pose additional major challenges in this task. The evaluation results presented in this article suggest that simple machine learning models created once at design time and then embedded into the structure for real-time querying can already yield acceptable load identification precision without having to rely on a finite element model of the structure to be available at runtime.

 ISBN: 978-1-61208-154-0

The remainder of this paper is organized as follows: Section II provides an overview of the basic AI approach towards creating sensorial structures. Section III presents a robust sensor network developed for reliable sensor data assessment and Section IV introduces a functional mockup system, which will serve to implement and evaluate different approaches to sensor data assessment and load case inference. An overview of two simple machine learning methods that were used for first trials with strain data from FEM simulation is given in Section V. Evaluation results are presented in Section VI. Section VII portrays how the intelligent agent paradigm can be employed for monitoring more complex structures in a distributed fashion. Section VIII closes with a conclusion and outlook.

II. An AI Approach towards Sensorial Structures

The AI-based process model for the design and realization of sensorial structures is depicted in Figure 1. First, a geometric CAD model of the desired technical structure is created. Based on the model, a library of standard load cases that are expected during operation as well as exceptional overloading or misuse cases is defined. This library can in principle comprise several hundred load cases, e.g., various forces and temperatures applied at various positions. FEM simulations then are run on the model, generating a large database of effect data (e.g., displacements, strains, stresses, temperature values etc.), which, depending on the complexity of the FEM model and the number of defined load cases, can be several gigabytes in size and may take several days to compute. It is, of course, impractical to directly run load case inference algorithms on such a large amount of precalculated data during actual operation of the structure, let alone to embed it into the structure using ROM circuits. Therefore, data compaction and data mining algorithms [7] [8] need to be applied to the FEM input and output in an offline preprocessing step to yield a considerably reduced dataset of key figures much fewer in count but of high informative value in terms of reliable load case estimation and overload detection. The simplest solution here would be to let a human expert choose selected sensor positions based on the geometric characteristics of the structure and manually rig the model with a desired distribution of sensors for assessing strain, pressure, temperature or similar effects that are present in the FEM calculation output. Mathematical optimization methods [9] [10] can be used to aid finding optimal sensor distributions at this stage. If a good sensor distribution has been found, a physical prototype equipped with real sensors is built, to which (a subset of) the previously defined load case library is physically applied in a series of training experiments. The sensor signals resulting from these load cases are recorded and stored along with the respective FEM results in the effects database. The same data reduction procedures that are performed on the FEM

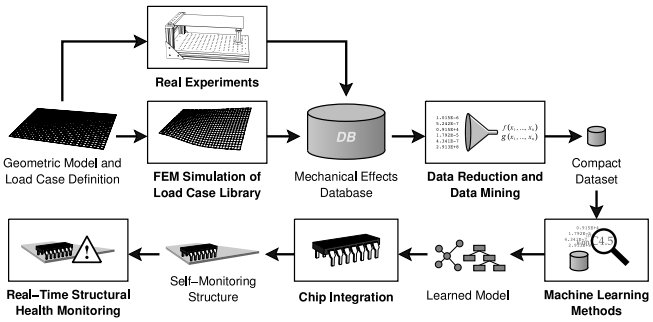


Figure 1. Artificial intelligence approach for the design of sensorial structures

output variables can be executed on the measured sensor values. For example, key figure calculation rules producing statistics (e.g., minimum, maximum, average value, standard deviation, or much more complex computations) over selected groups or multilevel hierarchies of sensors could be defined or automatically learned using machine learning and data mining techniques. It is an open research topic how such calculation rules can be automatically generated from the effects and sensor value database in the most beneficial way with respect to a given load case inference problem.

The resulting compact dataset is fed as training data into suitable machine learning algorithms [8] [11] [12]. The focus in this step lies on algorithms whose learned models as well as respective model evaluation procedures are simple and small enough to be directly transformed into miniaturized low-power hardware circuits. Embedding these circuits into the material yields a sensorial structure, which is able to constantly monitor its loading state during operation and issue alerts to its user when overloading is detected.

III. A Robust Sensor Network

When building structures that monitor their own health state by means of spatially distributed sensors, it is important that the introduction of local defects in the communication links between evaluation nodes, e.g., caused by overloading or misuse, does not result in entire regions of the structure being cut off from communication and, consequently, global monitoring processes. Hence, the communication infrastructure for sending sensor values and issued alerts across the structure must be robust with respect to sudden failure of communication links. For this purpose, we have implemented a sensor network capable of dynamically routing data packets across irregular sensor node topologies, e.g., caused by damage of communication links. Such an irregular network topology is shown in Figure 2. Network nodes are numbered from 0 to 12; communication links are depicted as arrows.

Each sensor node in the network processes the signals of a small local group of physical sensors (e.g., two to eight in count) and is connected to spatially neighboring nodes

 ISBN: 978-1-61208-154-0

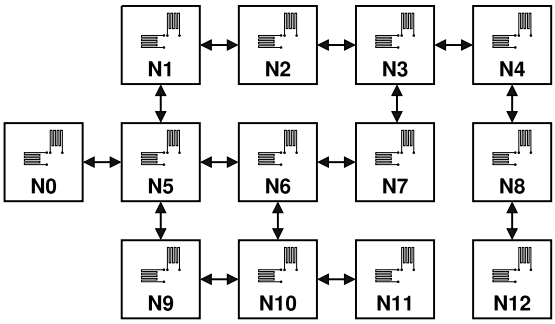


Figure 2. Irregular sensor network topology

via point-to-point serial links that define the network topology. A node provides analog signal conditioning, analog-to-digital conversion (ADC), and digital data processing. To enable the usage of unmatched and uncalibrated sets of sensors, the signal processing is implemented with a zooming window ADC approach. Spatial proximity relations among the physical sensors are directly translated to proximity relations in the network topology. With respect to communication, all nodes simultaneously act as routers and communication endpoints, with unique topological coordinates assigned to each of them. Communication is message based using the Simple Local Intranet Protocol (SLIP) [13], which employs dynamic routing strategies to forward data packets from their source nodes to desired destination nodes in the network. For this purpose, each packet contains a discrete multi-dimensional vector specifying the packet's destination in terms of remaining distance in the network topology. All routing information is held in the packets without any additional routing tables maintained by the sensor nodes. The SLIP communication protocol stack is implemented in hardware as a System-on-a-Chip architecture using FPGA/ASIC target technologies, and in software using the C and ML programming languages. Both Hardware and software implementation are automatically generated from the same high-level programming model using the ConPro synthesis framework [14]. The communication protocol is freely scalable in the network's size and topological dimensionality.

IV. Functional Mockup System

To practically evaluate the described approach, a functional mockup system was developed, which allows to experiment with different sensor assessment, data mining, and machine learning methods. As an evaluation scenario, we use a simple plate mounted with a fixture along one short side and a movable single point support at the bottom face. Different weights can be placed on the top face. The objective is to estimate the locations and masses of the weights and recognize overload situations only by examining a few strain values measured at the plate's bottom face using machine learning algorithms. For this purpose, pairs of orthogonally aligned strain gauges are attached to the bottom face of the plate at selected positions. Each pair measures the strain in the X and Y direction of the local face coordinate system and is connected to a miniaturized sensor node as described in Section III. The physical framework built for this scenario permits the installation of plates with different dimensions, material properties, and sensor distributions. Each possible sensor configuration consists of sensor nodes arranged in a topologically two-dimensional mesh network with each node having direct connections to up to four neighbors.

Initial trials using a nitrile rubber plate with glued-on commercial constantan strain gauges showed that the high load-induced strains occurring in the material lead to acceptable signal levels and satisfactory load identification results. Since the long-term objective is to integrate the sensors and electronic circuits into load-bearing structures, comparable load identification accuracy also needs to be attained for other common materials such as steel and for application cases with considerably smaller load-induced strains during standard operation—as is the case with steel loaded within its elastic range.

Figure 3 shows the experimental setup equipped with a St37 steel plate, on which four pairs of Ag strain gauges have been printed in an Aerosol Jet® printing process [15] using commercial pure silver ink. With material combinations permitting the increase of sintering temperatures, a future switch to CuNi alloys, which show significantly lower temperature dependence of resistivity, is envisaged. In the depicted setup, insulation between the printed conductive paths and the steel substrate was provided by a silicone-based coating layer, once more applied by means of Aerosol Jet® printing. However, in measurement tests with this plate and the sensor nodes' ADC components, the signal quality attained with the printed Ag strain gauges was not yet suitable for a reliable identification of loads limited to the elastic range.

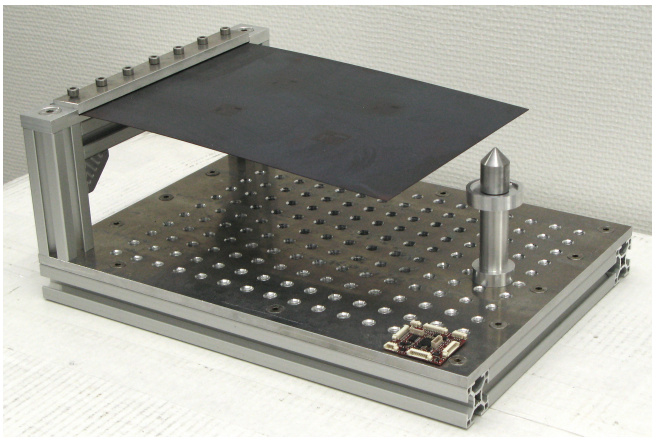


Figure 3. Metal plate setup

 ISBN: 978-1-61208-154-0

Hence, with a revision of the physical metal plate setup pending, the evaluation presented in this paper uses completely noise-free strain values from the FEM simulation step (cf. Figure 1) to gain a first set of reference success rates for the model of a $200 \times 300 \times 1$ mm St37 steel plate. For each examined machine learning method, the obtained results represent the best-possible prediction accuracy that theoretically could be achieved with (physically nonexistent) perfect sensors and ADC components. As the prediction accuracy can be viewed as an indicator for the general practicability of our approach, the success rates obtained in this evaluation will be used as benchmarks for further improvement of the hardware prototype (such as ADC components, sensor printing processes etc.) as well as development of more sophisticated machine learning approaches.

The FEM load case library used in the evaluation is defined by dividing the top face of the plate into a 20×20 mm grid of 176 equidistant load positions. At each position weights of 100 g and 200 g are separately applied, resulting in 352 different load cases with a single weight each. As the sensor inputs directly depend on the strains occurring in the material, overloading is defined in terms of displacements instead of strains. This adds a further complication to the evaluation scenario: We aim at investigating how good the tested machine learning algorithms perform at expressing correlations between different types of mechanical effects. Thus, overloading is defined to occur if and only if at least one point on the plate's surface moves downward in global Z direction more than a given threshold t_z. To prevent a bias towards positive or negative classifications in the training set, which may cause misleadingly higher success rates in the cross-validation examined in Section VI, the load case library is partitioned into 50 % overload and 50 % non-overload situations. This is done by obtaining the maximum Z displacement for each load case from the FEM output and setting the threshold to the median of these 352 values, yielding $t_z = 130.6\ \mu m$ for this particular scenario. Here, the desired result from the machine learning stage is a—preferably simple—mapping from the measured sensor values to estimated X and Y coordinates of the applied load (in the local surface coordinate system), an estimated mass of the weight in grams, and a yes/no classification whether the displacement threshold t_z has been exceeded by any point of the plate. Accordingly, the desired physical result is an actual low-power microchip that implements this mapping in hardware.

V. k-Nearest-Neighbor and Decision Tree Learning

Adhering to the design constraints imposed by the application scenario, experiments were started with an implementation of two of the most basic classification and regression algorithms. Namely, C4.5 decision tree learning [11] and the simple k-Nearest-Neighbor algorithm [12] were examined.

The training set contains all correct load positions, masses and classification results for every load case in the load case library along with the respective 8-dimensional strain vector determined via either sensor measurement or FEM simulation. To answer a query with k-Nearest-Neighbor (k-NN), the k entries closest to the query strain vector are selected and combined into a single estimation for the target variables. Numerical regression of the load position and mass can be achieved by calculating the weighted average, whereas discrete (e.g., Boolean) classification is possible by means of weighted voting. For a fixed k, let $E_1, \ldots, E_k$ denote the k nearest points in the training set, (x_i, y_i, m_i) the load coordinates and mass of E_i, and $Q \in \mathbb{R}^8$ the query vector. The estimated load position and mass (x_Q, y_Q, m_Q) for Q is calculated using inverse distance weighting, i.e.,

$$\begin{pmatrix} x_Q \\ y_Q \\ m_Q \end{pmatrix} = \sum_{i=1}^{k} w(E_i) \cdot \begin{pmatrix} x_i \\ y_i \\ m_i \end{pmatrix} \quad \text{with } w(E_i) = 1/(\mathrm{dist}^2(Q, E_i) \cdot w_{\mathrm{sum}}), \quad w_{\mathrm{sum}} = \textstyle\sum_{i=1}^{k} 1/(\mathrm{dist}^2(Q, E_i)) \tag{1}$$

and $\mathrm{dist}^2(\mathrm{Q}, \mathrm{E_i})$ being the squared Euclidean distance between points Q and E_i in 8-dimensional strain space.

Weighted voting is done by adding the weighting factors $w(E_i)$ of all E_i that have the same classification and then assigning to Q the classification that gained the largest sum, defaulting to a "safe" answer (yes) in case of a tie.

In our first experiments, the C4.5 decision tree learning is employed for classification only as its reliable extension to multivariate numerical regression is more elaborate.

VI. Evaluation Results

Since only values from FEM simulation are used in the evaluation, i.e., no comparison with measured values from the physical prototype is made, the spike placed under the plate's bottom face was modeled as a roller support in the FEM model for sake of simplicity. For C4.5 and k-NN with $1 < k < 10$, a leave-one-out cross-validation (LOOCV) was conducted on the training set described in Section IV: Each singleton subset of the training set was used exactly once for querying the machine learning models constructed from its complementary set and comparing the result with the known correct values. In addition, the models obtained from the entire 352 element training set were queried using an intermediate test set INT consisting of 150 g weights placed exactly at the midpoints between the previously defined grid positions, resulting in further 10×15 load cases neither the coordinates nor masses of which appear anywhere in the training set. Depending on the value chosen for k, the percentage of correct classifications achieved with k-NN varies between 89.77 % and 90.91 % in LOOCV and between 92 % and 93.33 % for INT. The C4.5 decision tree attained 94.32 % correct classifications in LOOCV and

 ISBN: 978-1-61208-154-0

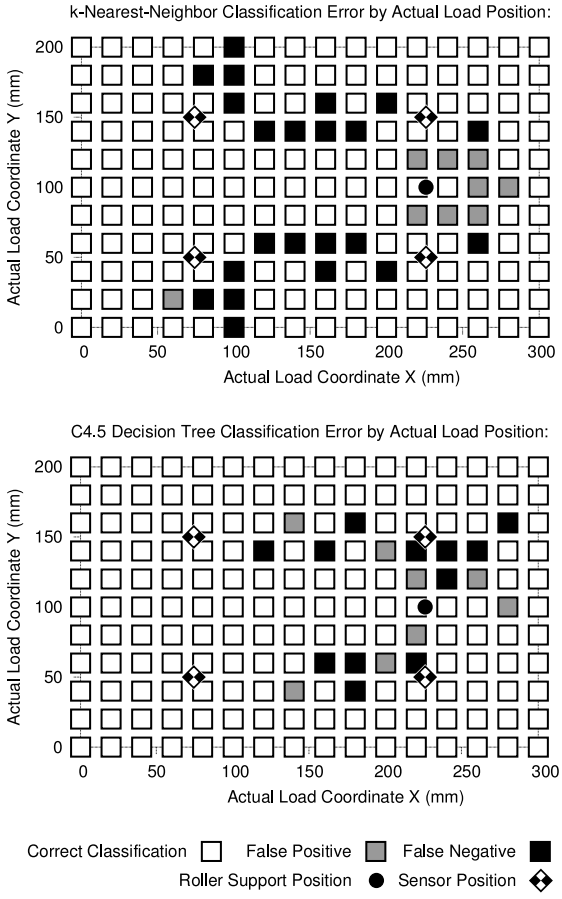


Figure 4. Classification results obtained with leave-one-out cross-validation on the training set

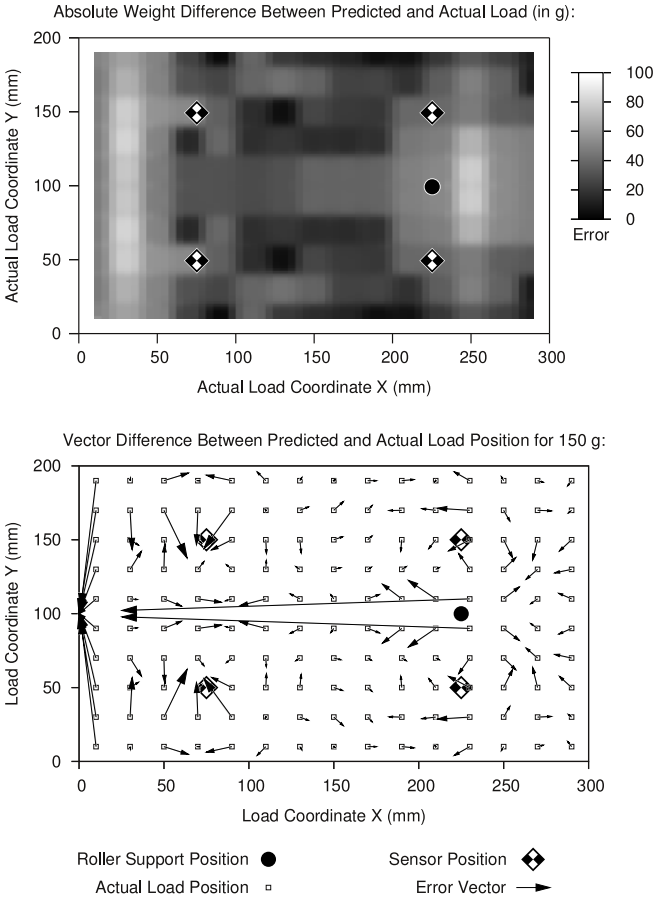


Figure 5. Regression error for 150 g weights placed at intermediate positions

96.22 % for INT. The locations where false positives and false negatives were returned in LOOCV by C4.5 and k-NN with $k = 4$ are shown in Figure 4. While at each position a 100 g and a 200 g load case were tested, no position produced wrong classifications for both masses.

The numerical regression results obtained for the test set INT by k-NN with $k = 4$ are illustrated in Figure 5. The difference vectors from the actual to the estimated load positions are shown in the bottom vector field (median length 9.09 mm, average length 15.88 mm), whereas the absolute error in load mass estimation is visualized in the upper plot (median 20.39 g, average 25.38 g). Among all tested values for k, the median difference vector length of the load coordinate regression was smallest with $k = 4$, while the average was smallest with $k = 2$. In Figure 4 and 5, the locations of the strain sensors and the roller support are indicated with diamond and circle symbols, respectively. The fixture is located on the Y coordinate axis. It is clearly noticeable that the regression error is largest in the vicinity of the supports.

VII. Monitoring More Complex Structures with Intelligent Agents

While the evaluation results suggest that the tested machine learning methods are, in principle, suitable for load case estimation on simple structures like a metal plate, it is not clear how these methods can successfully be utilized for monitoring much more complex technical structures as well, e.g., objects that comprise many, possibly movable, parts. Several problems arise in such application scenarios. Firstly, the definition of a comprehensive and adequate load case library is more difficult for these structures as the number of relevant load cases might increase exponentially with the number of constituent substructures and the mechanical effects might depend on the movable parts' positions and orientations. Secondly, even if the load case library or FEM effects database does not already reach a prohibitive size due to combinatorial growth, the more complicated the cause-effect relations reflected in the produced datasets become, the less fruitful an identification and extraction of these patterns into a proper model may prove in the data mining and machine learning stages. As a consequence, higher error

 ISBN: 978-1-61208-154-0

rates are to be expected in the actual load case inference. Finally, the machine learning model for the entire structure may grow to a size where it neither can be created or queried as a whole in reasonable time with the available resources nor fit on a single chip. To circumvent these difficulties, we propose an intelligent agent based approach to solving the global monitoring problem by splitting it into multiple spatially constrained subproblems and solving these in a distributed fashion. Such multi-agent systems have already been successfully applied to acoustic detection of high-velocity impacts [4].

Generally, an agent can be viewed as a hardware and/or software entity that receives perceptions from an environment it is situated in and reacts to them, either reflex-based [16] or deliberately [17], in order to perform one or more specific tasks [18]. Applied to the task of monitoring a sensorial structure, the agent may be physically situated on the latter, with its perceptions coming from different groups of sensor nodes and communication with other agents. If the agent has access to a knowledge base of facts about the structure's geometric and mechanical characteristics as well as cause-effect relations with respect to load introduction, it can draw conclusions about the current state of its environment based on its sensory input, i.e., make sense of the sensor signals. It is the presence of this semantic level that distinguishes intelligent agents from nodes of the sensor network described in Section III: While the latter only provide the infrastructure for routing measured sensor data across the physical object and have no way of interpreting this data, the agents are capable of incrementally constructing a mental model of their surroundings over time.

In our approach [19], the structure is partitioned into connected substructures, which can be further partitioned into regions. To each region an individual monitor agent is assigned that regularly adjusts its internal model of the respective region according to its perceptions. Communication among agents covering directly neighboring regions enables the construction of a (simplified) distributed global view of the entire structure. For this purpose, each monitor agent requires an appropriate description of how loads and their effects are transmitted across the boundaries between the agent's region and each of the adjacent regions. That way, an agent can infer from its local sensory input the structural state at these boundaries and send the results to its topological neighbors, which in turn update their local models by combining their own sensor data with the boundary state information they received from their respective neighbors, and, again, pass the updated boundary states on to these, and so forth. Establishing a multilevel hierarchy of agents and incorporating (to a manageable extent, given the available resources) equation-based knowledge from the FEM domain into the reasoning process may yield a distributed load inference algorithm that is adaptive in terms of desired temporal resolution and predictive precision, depending on the number of communication rounds performed per time unit. Thus, one of the next logical steps in our research will be the development of such algorithms along with the necessary agent communication protocols and their evaluation in the functional mockup system.

VIII. Conclusion and Future Work

This article presented a novel AI-based process for the design of self-monitoring sensorial structures that utilizes machine learning methods for resource-constrained real-time load case inference. A distributed sensor network architecture and a functional mockup system using a simple steel plate as evaluation scenario was introduced. The general practicability of the machine learning approach was shown for this scenario using noise-free data from FEM simulation only. In the evaluation very simple algorithms like k-NN and decision tree learning already yielded over 90 % correct classifications in the detection whether the plate's maximum displacement vector exceeded a given length due to load introduction. Also numerical regression of the load locations and masses with k-NN attained a prediction quality that may be acceptable in many practical application cases. It seems likely that the results can be improved by employment of more advanced machine learning methods and, in particular, by incorporation of specific domain knowledge from the field of applied mechanics into these. The focus here should lie on the best possible elimination of false negatives, which correspond to unrecognized overload situations, and on the increase of prediction quality for sensor signals with a low signal-to-noise ratio. On the hardware side, further improvement of functional printing processes as well as miniature ADC components is expected to lead to better signal quality and, thus, more accurate load identification results. In addition to this, one of the next steps in our research will be the development and implementation of multi-agent based approaches that enable monitoring of geometrically much more complex structures. When the viability of this has been shown, the machine learning models need to be enhanced to automatically accommodate to structural aging while in operation, e.g., in materials like polymers or textiles, where this aspect is of particular importance.

Acknowledgment

The authors gratefully acknowledge the financial support by the State and University of Bremen in the framework of the ISIS (Integrated Solutions in Sensorial Structure Engineering) Sensorial Materials Scientific Center (fund numbers 54416906, 54416918, and 54416919).

References

[1] M. Maiwald, C. Werner, V. Zöllmer, and M. Busse, "INKtelligent Printing® for sensorial applications," *Sensor Review*, vol. 30, no. 1, pp. 19–23, 2010.

[2] D. Mayer, T. Melz, C. Pille, and F.-J. Wöstmann, "CAST-TRONICS – direct integration of piezo ceramic materials in high pressure die casting parts for vibration control," in *Proceedings of Actuator 2008 – 11th International Conference on New Actuators*, Bremen, Germany, June 9th-11th 2008, pp. 61–66.

[3] W. Lang, F. Jakobs, E. Tolstosheeva, H. Sturm, A. Ibragimov, A. Kesel, D. Lehmhus, and U. Dicke, "From embedded sensors to sensorial materials – the road to function scale integration," *Sensors and Actuators: A Physical*, 2011, doi:10.1016/j.sna.2011.03.061.

[4] W. H. Prosser, S. G. Allison, S. E. Woodard, R. A. Wincheski, E. G. Cooper, D. C. Price, M. Hedley, M. Prokopenko, D. A. Scott, A. Tessler, and J. L. Spangler, "Structural health management for future aerospace vehicles," in *Proceedings of the 2nd Australasian Workshop on Structural Health Monitoring*, Melbourne, Australia, 2004.

[5] V. K. Wadhawan, *Smart Structures: Blurring the Distinction between the Living and the Nonliving*. Oxford Univ. Press, 2007.

[6] R. J. M. Vullers, R. van Schaijk, I. Doms, C. Van Hoof, and R. Mertens, "Micropower energy harvesting," *Solid State Electron.*, vol. 53, no. 7, pp. 684–693, 2009.

[7] O. Maimon and L. Rokach, Eds., *The Data Mining and Knowledge Discovery Handbook*. Springer, 2005.

[8] A. J. Izenman, *Modern Multivariate Statistical Techniques: Regression, Classification, and Manifold Learning*. Springer, 2008.

[9] R. Jedermann and W. Lang, "The minimum number of sensors - interpolation of spatial temperature profiles," in *Proceedings of the 6th European Conference on Wireless Sensor Networks (ESWN)*, U. Roedig and C. Sreenan, Eds. Springer, 2009, pp. 232–246.

[10] A. R. M. Rao and G. Anandakumar, "Optimal placement of sensors for structural system identification and health monitoring using a hybrid swarm intelligence technique," *Smart. Mater. Struct.*, vol. 16, no. 6, pp. 2658–2672, 2007.

[11] J. R. Quinlan, *C4.5: Programs for Machine Learning*. Morgan Kaufman Publishers, Inc., 1993.

[12] T. M. Mitchell, *Machine Learning*. McGraw-Hill, 1997.

[13] S. Bosse and D. Lehmhus, "Smart communication in a wired sensor- and actuator network of a modular robot actuator system using a hop-protocol with delta routing," in *Proceedings of Smart Systems Integration 2010 - 4th European Conference and Exhibition on Integration Issues of Miniaturized Systems*, T. Gessner, Ed., Como, Italy, March 23st-24rd 2010, CD-ROM.

[14] S. Bosse, "Hardware-software-co-design of parallel and distributed systems using a unique behavioural programming and multi-process model with high-level synthesis," in *Proceedings of SPIE Microtechnologies Conference 2011, Session EMT 102 VLSI Circuits and Systems*, Prague, Czech Republic, April 18th-21th 2011.

[15] M. Maiwald, C. Werner, V. Zöllmer, and M. Busse, "INKtelligent® printed strain gauges," in *Proceedings of the Eurosensors XXIII conference*, Lausanne, Switzerland, September 6th-9th 2009, pp. 907–910.

[16] R. A. Brooks, "Elephants dont play chess," *Robot. Auton. Syst.*, vol. 6, no. 1&2, pp. 3–15, 1990.

[17] A. S. Rao and M. P. Georgeff, "Decision procedures for BDI logics," *J. Logic Comput.*, vol. 8, no. 3, pp. 293–343, 1998.

[18] N. R. Jennings, "On agent-based software engineering," *Artif. Intell.*, vol. 117, no. 2, pp. 277–296, 2000.

[19] M. Lawo, H. Langer, D. Lehmhus, M. Busse, A. Burblies, and W. Lang, "Simulation techniques for the description of smart structures and sensorial materials," *J. Biol. Phys. Chem.*, vol. 9, pp. 143–148, 2009.

 ISBN: 978-1-61208-154-0

A Soft Case-based Reasoning System for Travelling Time Estimation

Lixing Wang
Department of Industrial and Systems Engineering
The Hong Kong Polytechnic University
Hong Kong
e-mail: lx.wang@polyu.edu.hk

Wai-Hung Ip
Department of Industrial and Systems Engineering
The Hong Kong Polytechnic University
Hong Kong
e-mail: mfwhip@inet.polyu.edu.hk

***Abstract*—Soft computing, which includes fuzzy logic, neural network theory, evolutionary computing and probabilistic techniques, is an emerging approach to computing. It resembles biological processes more closely than traditional techniques. Case-based Reasoning (CBR), which normally includes four different phases in the problem-solving cycle, is an effective methodology for solving these kinds of problems. Integration of soft computing techniques into a CBR system can significantly help people to solve problems with more accuracy. This study aims to develop a soft CBR System using fuzzy logic and neural network theory to estimate travelling time of vehicles. A simulated case is also studied to test the performance of the system. The results show that the soft CBR system is able to achieve accurate solutions.**

Keywords-Case-based Reasoning; Soft Computing; Fuzzy Logi;, Neural Network; Travelling Time.

I. INTRODUCTION

Vehicle travelling time is an important factor in logistics. Accurate prediction of travelling is very helpful both to goods delivery and to make vehicle schedules [4]. When vehicles are used to deliver consignments, it is necessary to give an accurate estimation of time they will take to reach the destination. It will be convenient to customers if the logistics company can supply an accurate arrival time. Customers can prepare to receive the consignments. Accurate transportation time estimation is also important for making schedules. Firstly, accurate estimation of time can help planner to know whether the consignment would arrive at the destination before the deadline. Furthermore, most optimization algorithms for vehicle routing problems are studied based on the information of vehicle travelling time. One challenge for the application of these optimization algorithms in industries is the accurate prediction of the arrival time of the consignments, since the travelling time is hard to predict for the logistics companies. The optimization results are also unfeasible if it is computed based on a inaccurate travelling time estimation. Consequently, logistics companies prefer making the schedules based on their experiences. Moreover, if some accidents which will affect the travelling time of vehicles happened, there is also a need to adjust the estimation of the travelling time.

This research aims to develop a soft CBR system to estimate the travelling time of vehicles. The system integrates soft computing techniques: fuzzy logic [11], neural network theory [5] and the concepts of CBR [1] together. With the help of the system, the travelling time can be estimated more accurate [5, 6]. This is because traditional CBR system can just forecast travelling time based on the cases in its database. But the conditions will change over time, such vehicle speed will increase, more vehicles will be on the road and so on. The traditional can just used outdated information to forecast, and it cannot achieve accurate results. The proposed system has the machine learning function. The system can update cases automatically. It can achieve more accurate results than traditional CBR system.

The rest of this paper is organized as follows: Section II contains a review of previous studies. In Section III, the system architecture is introduced. How integrate CBR and soft computing is also discussed in this section. In Section IV, a simulated case is computed to illustrate the system and to test the performance of the system. The final section draws conclusions from the research. The future work on this topic is also discussed.

II. LITERATURE REVIEW

Case-based Reasoning (CBR) is a designed model for expert systems [1]. It focuses on the reuse of experience [2]. Aamodt and Plaza [3] describe the traditional CBR model, which defines the problem-solving cycle in four different phases: Retrieval, Reuse, Revise and Return. Figure 1 illustrates the cycle.

The problem solving life cycle in a CBR system consists essentially of four parts:

1. RETRIEVE the most similar case or cases
2. REUSE the information and knowledge in that case to solve the problem
3. REVISE the proposed solution
4. RETAIN the parts of this experience likely to be useful for future problem solving

A new problem is solved by retrieving one or more previously experienced cases, reusing the case in one way or another, revising the solution based on reusing a previous case, and retaining the new experience by incorporating it into the existing knowledge-base (case-base).

An initial description of a problem (top of figure) is regarded as a new case. It is used to RETRIEVE a case from the collection of previous cases. The retrieved case is combined with the new case - through REUSE - into a solved case, i.e. a proposed solution to the initial problem. Through the REVISE process this solution is tested for success, either application to the real world environment or evaluation by an expert, and repaired if failed. During

 ISBN: 978-1-61208-154-0

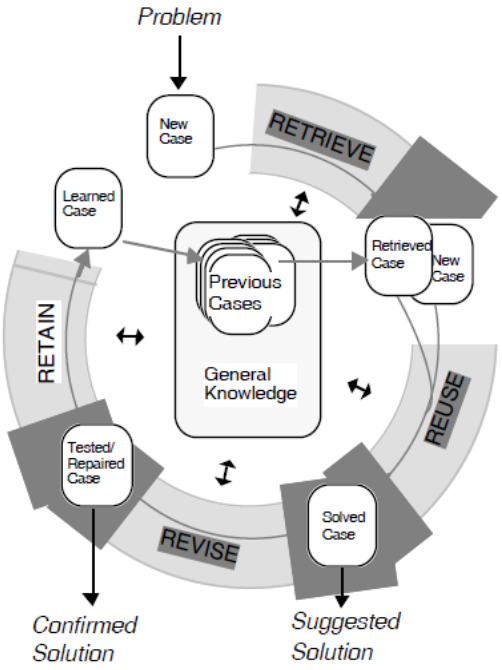


Figure 1. Case-based Reasoning Cycle (derived from [3])

RETAIN, useful experience is retained for future reuse, and the case base is updated by a new learned case, or by modification of some existing cases.

Sadek et al. [4] have successfully developed a prototype case-based reasoning system for real-time freeway traffic routing. CBR can solve new problems by reusing solutions of similar past problems. The result of their research has demonstrated that the prototype system can run in real-time and produce high quality solutions using case-bases of reasonable size. Shen et al. [5] proposed an approximate CBR model. This model uses the neural networks technology to process fuzzy inference with the two dualities of fuzzy logic and approximate reasoning. The self-organizing and self-learning procedure can be executed by modifying the weight. Maria and Maite [6] proposed retention and forgetting strategies to maintain the case base of the CBR system a certain scale by adding and removing cases. It has been proved in their research that the model they proposed can maintain the case base effectively. Anthony and Xun [7] successfully applied CBR system to handle planning applications in development control. The system helped the planner to reuse previous similar cases in making decision on the new applications. Castro et al. [8] developed a fuzzy system to improve CBR system in solving risk problems. Some rules are developed in the fuzzy system to search for the most suitable case not the most similar case. Passone et al. [9] incorporate domain-specific knowledge into a genetic algorithm to implement CBR adaptation. The research improved the adaptation phase in the CBR system. The improved system is suitable to deal with numerical modeling applications that require the substitution of a large number of parameter values.

These successful applications and relevant approaches encourage us to further research about CBR system for our problem: travelling time estimation. A soft CBR system has been developed. Fuzzy logic and neural network theory are applied in the system. In the next section, details about the system will be discussed.

III. METHODOLOGY

Figure 2 shows the system architecture used in this study. The system includes two parts. The first part separates the planned route of the vehicle into several segments. It helps users to find the same route from the case base. If there is not the same route, the most similar route will be selected using fuzzy logic and CBR. The other part of the system uses CBR to calculate the time of each segment of the planned route. The weightings of the CBR are trained using Neural Network theory. Finally, the case base is updated using some rule-based strategies.

A. Route Division

Firstly, route division part of the system will be introduced. When the user inputs the start location and the destination to the system, several routes will be generated. The user can choose one as the planned route. Then the system can help the user to estimate the time that needed for the vehicle from the start point to the destination. The first step is to divide the planned route of the vehicle into several segments using important traffic cross points. The Figure 3 shows the route segments of an example. If the vehicle is transported from point 1 to point 9, the planned route is 1-2-8-9. The system divides the route into three parts: 1-2, 2-8, 8-9. Then, it searches the case database for the same segments.

If the system cannot find the same route segment from the case base, the most similar route will be selected using fuzzy logic. Since road grade and city scale are two main factors to affect the vehicle speed, the fuzzy sets in this system include road grade (1, 2, 3 and 4), which indicates the designed vehicle speed on that road and city scale (super, big, middle and small), which indicates the population in that city. Figure 4 shows the membership functions. The sum of each similarity measure is the finally similarities of the route. The route with the biggest value is the most similar route.

There are two reasons for dividing a route into several segments. One is that the short route segments have relatively more similar routes than long route segments which can reduce the scale of the case base, so the search speed can also be increased. The other reason is that the number of same routes will be increased. Two vehicles may start from different places and also arrive at different places, but part of their routes may be the same.

B. CBR System

CBR sub-system is the core of the soft CBR system. The module is mainly designed using CBR. The weight coefficients of all factors which affect the process of degeneration are saved in a database. The design of the database is shown in Table I.

 ISBN: 978-1-61208-154-0

Different kinds of data have different weightings (w_i). The weightings need to satisfy the constraints:

$$\sum_{i=1}^{n} w_i = 1 \quad (1)$$

where, $0 \le w_i \le 1$ (i = 1, 2… …n)

When the vehicles finish their transportation tasks, they will give the information listed in the table to the backend system. The system will store the data in the case database and produce a new case_id for it. The case database is shown in Table II.

When the system begins to estimate the travelling time, it firstly searches for the case base. If one kind of data of the vehicle matches the same kind of data of a case, the system will record 1 as the value of x_i in the blank space of match_degree. Then, it multiplies the weighting of this kind of data by the match_degree ($w_i \times x_i$) to produce the result. After calculating all the types of data belonging to the case, the system adds all the results together. The sum (M) is the degree to which the case matches the problem that needs to be solved.

$$x_i = \begin{cases} 0, & \text{not match,} \\ 1, & \text{match,} \end{cases} \quad (2)$$

i = 1, 2……n,

$$M = \sum_{i=1}^{n} w_i x_i \quad (3)$$

where, M indicates the degree to which the case matches the problem.

After calculating all the cases of the same route, the system chooses the best match in the database and then uses the time consumed on this case as the predicted result of this segment. All the time needed by each segment added together is the time for the vehicle to arrive at the destination.

On the other hand, if all the degree to which the case matches the problem is less than 0.7 (the value is a coefficient and users can adjust it based on real conditions), the system will compute the deviation of the most existing similar route segments under the condition of the best match case and the

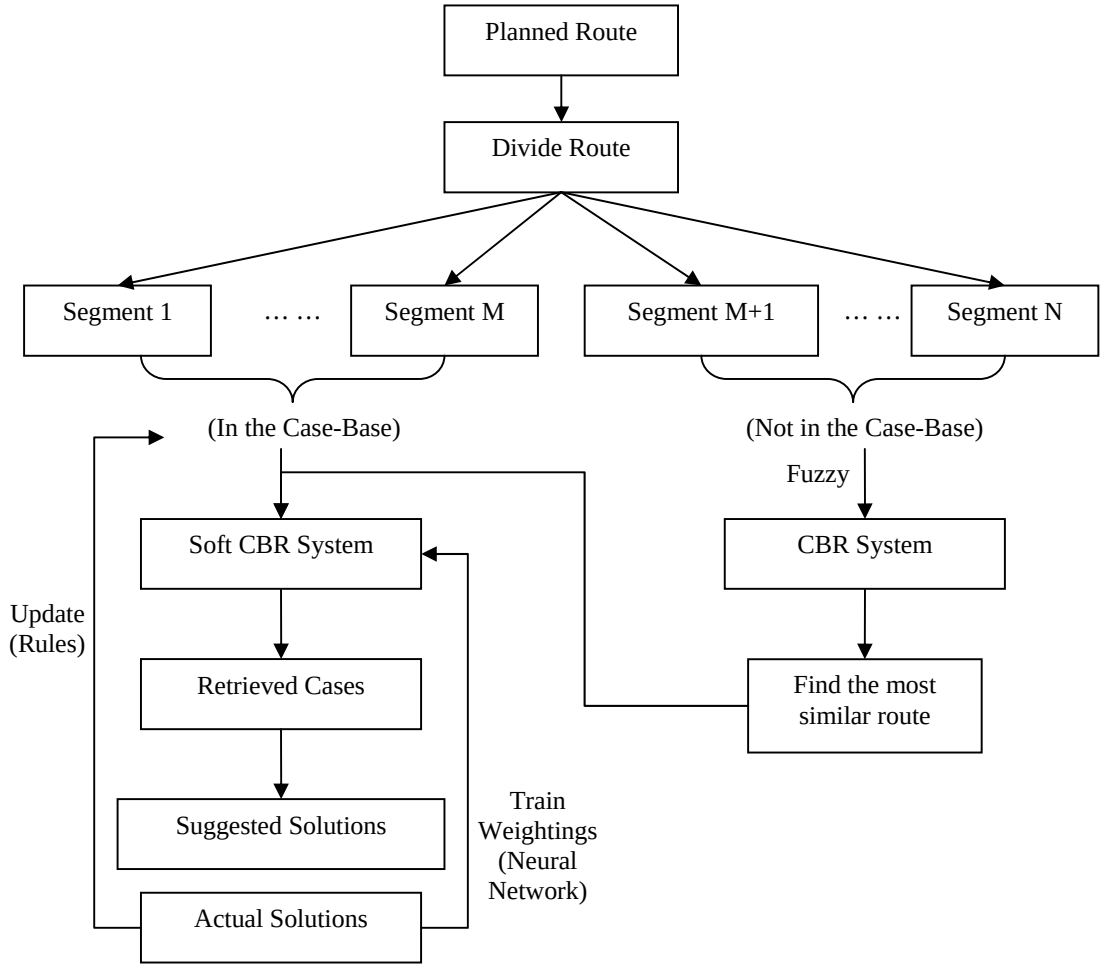


Figure 2. System architecture

 ISBN: 978-1-61208-154-0

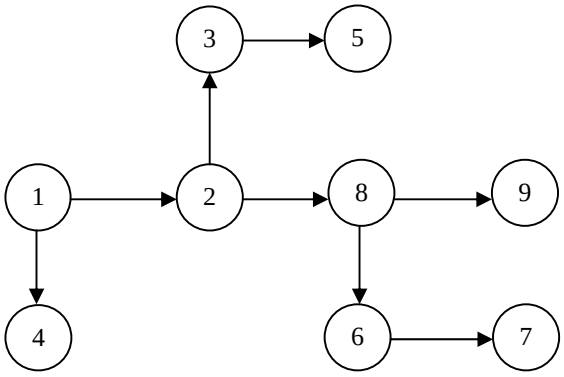


Figure 3. An example of route separation

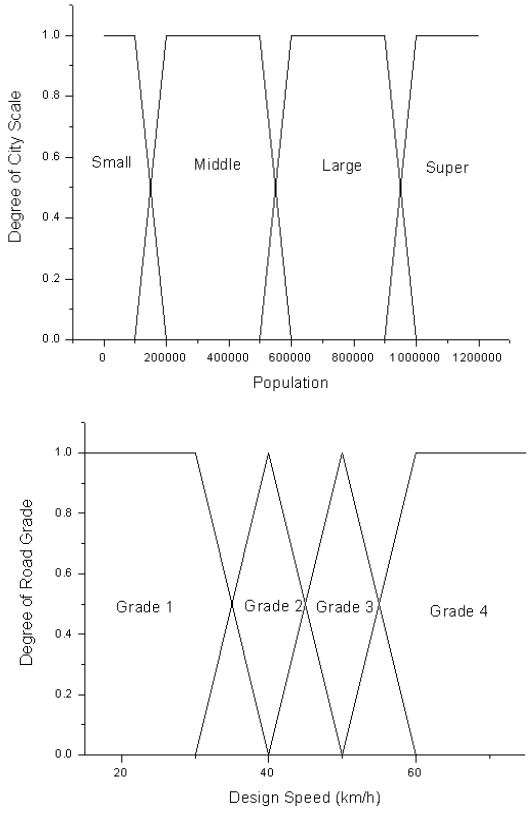


Figure 4. Membership Functions

condition of the unsolved problem. The deviation ratio will be set as the deviation ration of this unsolved problem and the best match case. Then the travelling time can be computed.

Finally, the total time spend for each segment is the time needed for consignments delivery.

In the CBR system, w_i is used to represent the weighting of each factor in a case for calculating the travelling time. Sometimes, the weightings in the database are not accurate. They are need to be adjusted. Which situations the system needs to adjust will be introduced in Part C of this section. And in this part, the adjustment methodology will be discussed firstly. Neuron network is applied to train these weightings. The details will be introduced as follows.

Let $X = \{x_1, x_2, ..., x_n\}$ be a set of n vectors, where the components of each vector represents the match degree of Case $_j$. w_i is the coefficient. The value is decided by specific segments. Different segments has different set of w_i . A simple single layer neural network is used to train their values [10].

Step 1: Initialization

Set initial weights w_i and threshold θ to random numbers.

Step 2: Activation

Activate the perceptron by applying inputs $x_i(p)$ and desired output $Y_d(p)$, which means the actual travelling time. Calculate the actual output at iteration *p*=1

$$Y(p) = step[\sum_{i=1}^{n} x_i(p) w_i(p) - \theta] \quad (4)$$

where *n* is the number of the perceptron inputs, and formula (4) is a step activation function.

Step 3: Weight training

Update the weights of the perceptron

$$w_i(p+1) = w_i(p) + \Delta w_i(p) \quad (5)$$

where $\Delta w_i(p)$ is the weight correction at iteration p.

The weight correction is computed by the delta rule:

$$\Delta w_i(p) = \alpha \times x_i(p) \times e(p) \quad (6)$$

$$e(p) = Y_d(p) - Y(p) \quad (7)$$

Step 4: Iteration

Increase iteration p by one, go back to step 2 and repeat the process until convergence.

Then w_i can be determined.

TABLE I. THE WEIGHTING COEFFICIENTS DATABASE

Weighting (w_i)	**Factor**	**Match_Degree (X_i)**
w_1	*Weather*	1/0
w_2	*Workday/Holiday*	1/0
w_3	*Time_Period*	1/0
w_4	*Vehicle_Type*	1/0
w_5	*Driver*	1/0
w_6	*Products Weight*	1/0
……	……	1/0
w_n	*Fuel_level*	1/0
	Sum	$\sum w_i x_i$

TABLE II. THE CASE DATABASE FOR MONITORED CONTAINERS

Case_ID	Time_Consumed (min)	Match_Degree
A01A0201	36	0.5
A02B0301	44	0.7
B03B0501	15	0.9
A02C0801	24	0.3
C08C0901	17	0.5
C08C0601	32	0.7

C. Updating Case Base

Everything is in course of continuous movement change, and development in the world. With the developing of vehicles and cities, the travelling time needed from the same start location and destination under the same condition will be different. The old cases in the database will be unsuitable for new arriving cases. Consequently, it is necessary to update the old case base, when the value of deviation is too big. The case base is updated using the rules listed below. Firstly, the system will check whether the segment calculated is the same route with the unsolved problem. If it is the most similar route, the actual segment and its result will be set as a new case in the case base. For the same route segment, if the new case can find a previous case from the case base, which $M = 1$, the old case will be replaced by the new case. If the results are different, the weighting of the case will be trained again. If $M \neq 1$, the case will also be added into the case base.

Rule 1: If the segment is the same as the actual segment and the match degree is not less than 0.7, then go to Rule 3;

Rule 2: If the segment is the same as the actual segment and the match degree is less than 0.7, then train the weighting and add the case into the case base.

Rule 3: If $M = 1$, then go to Rule 5

Rule 4: If $M \neq 1$, then train the weighting using neural network and add the case into the case base.

Rule 5: If (Result_New-Result_Old) / Resul_Old >5% or (Result_New-Result_Old) / Resul_Old < -5%, then train the weighting of this segment and add the case into the case base.

IV. CASE STUDY

In this section, a case study is used to illustrate how to apply this system to calculate the travelling time. Since it is difficult to collect adequate data from real transportation system, a simulated case is studied to describe the operating procedure of the system.

YT is a simulated logistics company in mainland China. The company plans to transport consignments from Shanghai to Beijing. A suggested route can be generated by the system based on Google map. The routes include: 311 Guangfu Road in Shanghai to G2 Highway, G2 Highway, S29 Highway, G25 Highway, G18 Highway, G3 Highway, S30 Highway, Jingjing Highway, Jingjing Highway to 14 Hepingli in Beijing. All the route segments can be found in the case base except the first one and the last one. The first one, the route from 311 Guangfu Road in Shanghai to G2 Highway, will be used to illustrate the operation of the Fuzzy part of the system. In contrast, another route G2 Highway, which can be found from the case base, will be used to illustrate the Neural Network part of the system as an example.

TABLE III. CASE INFORMATION

Weather	Sunny
Workday/Holiday	Workday
Time_Period	8:45
Vehicle_Type	Truck_JF_15T
Driver	00012
Products Weight	15T

TABLE IV. CASE INFORMATION

Weighting (W_i)	Case_2314	Match_Degree (X_i)
0.20	*Weather*	1
0.30	*Workday/Holiday*	1
0.30	*Time_Period*	1
0.05	*Vehicle_Type*	1
0.05	*Driver*	1
0.10	*Products Weight*	1
	Sum	1

How the system calculates the time spends in the first route segment will be introduced. The fuzzy sets of this case are city scale and road grade. The city scale is measured by the population, which is more than 10 million. And the designed road grade is grade 2. Through computing by fuzzy logic, the most similar route segment is found. Then the system sets the most similar route segment as the unsolved new case and input it into the CBR system. Considering weather, driver and other factors listed in table I, the most similar case can be found. Then, formula (8) can be used to estimate the travelling time of this route segment. The method is the same as the calculation of the travelling time for the segment: G2 Highway.

$$t = t_s \cdot \frac{L}{L_s} \tag{8}$$

where, t_s is travelling time of the similar case, L_s is the route length of the similar case. t is the travelling time of the unsolved route segment. L is the route length of the unsolved route segment.

For the route segment that can be found in the case base, G2 Highway is used to illustrate. The G2 Highway in this case is shown in the table III. The most similar case that $M = 1$, Table IV shows the details. Since the $M = 1$, the travelling time of this similar case is regarded as the estimated time of this route segment, When all the segments are calculated, the time added together will be the total estimated time. After simulation, the division of real travelling time and the estimated result is bigger than 5%. Consequently, the weightings of this case need to be adjusted. Neural network is applied. After adjustment, the case base is updated using the new result of the G2 Highway.

 ISBN: 978-1-61208-154-0

V. CONCLUSION AND FUTURE WORK

The main contribution of the study is to propose a soft case-based reasoning system to estimate travelling time of vehicles. The system is developed based on the traditional CBR system. It integrates Fuzzy logic and neural network techniques. Consequently, the system has the machine learning function. It can update itself automatically. The system can firstly divide the planned route into several segments. Then it helps users to search for the most similar case with each segment. If the deviation of the estimated time and real travelling time is huge, the weightings will be trained using neural network and old case will also be replaced. The system combines soft computing and case-based reasoning techniques to estimate travelling time. With the help of the system, the arrival time can be predicted more accurately. A simulated case is used to test the system. The results showed the system can be applied to estimate the travelling time. The accuracy increased with the help of neural network and fuzzy logic after some training. However, it is difficult to collect adequate data from real transportation system. It is hoped the system will be applied in the real transportation system in the future, and then its performance can be evaluated.

ACKNOWLEDGMENT

The authors wish to thank the Research Committee of The Hong Kong Polytechnic University for support in this research work.

REFERENCES

[1] I. Watson, “Applying Case-Based Reasoning”, Morgan Kaufmann, San Francisco, CA, 1997.

[2] D. W. Aha, “The omnipresence of case-based reasoning in science and application”, Knowledge-Based Systems, vol. 11, pp. 261–273, 1998.

[3] A. Aamodt, and E. Plaza, “Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches”, AI Communications vol. 7, pp. 39-59, 1994

[4] A. W. Sadek, B. L. Smith, and M. J. Demetsky, “A prototype case-based reasoning system for real-time freeway traffic routing”, Transportation Research Part C, vol. 9 pp. 353-380, 2001.

[5] Z. Shen, H.C. Lui, and L. Ding, “Approximate Case-Based Reasoning on Neural Networks”, International Journal of Approximate Reasoning, vol. 10, pp. 75-98, 1994

[6] S. Maria, and L.S. Maite, “Adaptive case-based reasoning using retention and forgetting strategies”, Knowledge-Based Systems vol. 24, pp. 230-247, 2011

[7] A.G.O. Yeh, and X. Shi, “Case-based reasoning (CBR) in development control”, JAG, vol. 3(3), pp. 238-251, 2001

[8] J.L. Castro, M. Navarro, J.M. Sanchez, and J.M. Zurita, “Introducing attribute risk for retrieval in case-based reasoning”, Knowledge-Based Systems, vol. 24, pp. 257-268, 2011.

[9] S. Passone, P.W.H. Chung, and V. Nassehi, “Incorporating domain-specific knowledge into a genetic algorithm to implement case-based reasoning adaptation”, Knowledge-based Systems, vol. 19, pp. 192-201, 2006.

[10] M. Negnevitsky, Artificial Intelligence: A Guide to intelligent systems (Second Edition) (Ch. 6 168-174),2005

[11] J. Ma, S. Chen and Y. Xu, “*Fuzzy logic from the viewpoint of machine intelligence*”, Fuzzy sets and systems, vol. 157, pp. 628-634, 2006.

 ISBN: 978-1-61208-154-0

Optimizing the Area Under the ROC Curve in Multilayer Perceptron-based Classifiers

Raúl Ramos-Pollán
Centro Extremeño de Tecnologías Avanzadas
CIEMAT
Trujillo, Spain
e-mail: raul.ramos@ciemat.es

Naimy González de Posada,
Miguel Angel Guevara López
INEGI-Faculty of Engeniering
University of Porto
Porto, Portugal
e-mail: {nposada,mguevaral}@inegi.up.pt

Abstract - **This paper proposes a method to adapt existing multilayer perceptron (MLP) training algorithms for optimizing the area under the receiver operating characteristic curve (AUC), in binary classification tasks. It is known that error rate minimization does not necessarily yield to optimal AUC and, rather than developing new MLP training algorithms for AUC optimization, we reuse the vast experience encoded into existing algorithms by replacing the error metrics used to guide their training processes, with a novel defined AUC loss function, leaving unmodified their core logic. The new method was evaluated over 2000 MLP configurations, using four different training algorithms (backpropagation, resilient propagation, simulated annealing and genetic algorithms) in 12 binary datasets from the UCI repository. AUC was improved in 5.86% in average and, in addition, the proposed definition preserves interesting properties of other error metrics. An efficient AUC calculation procedure was also developed to ensure the method remains computationally affordable.**

Keywords - Multilayer perceptron, AUC optimization, error measure, machine learning, binary classifiers.

I. INTRODUCTION

Receiver Operating Characteristic (ROC) analysis [1] was originally used in signal processing and, more recently, is commonly used in biomedicine [2]. ROC curves measure the capability of a binary test or classifier to correctly distinguish between positive and negative instances, accounting for the trade-off between true and false positives rates, and the area under the ROC curve (denoted by AUC or Az) is used as a scalar comparative metric. In medicine, ROC curves are used to analyze and compare diagnostic systems [3] or for mining biomedical data [4]. In machine learning they are increasingly used to evaluate and compare classifier performance in general [5, 6].

Machine learning classifier performance is often measured by its accuracy (number of dataset elements correctly classified), which is obtained by fixing a specific threshold on a score produced by the classifier for each dataset element. Both accuracy and AUC are complementary measures to evaluate classifier performance, and AUC has the property of being insensitive to class distribution (class skew). Although both are obviously related, it is well established that error rate minimization does not necessarily yield AUC optimization [7], and several efforts have been invested into using AUC in machine learning [6] and, most recently, to build AUC optimizing classifiers, mostly from scratch or by re-designing the core of existing algorithms to include AUC metrics, such as support vector machines [8], gradient descent [9], evolutionary algorithms [10] and others [11-16].

In a previous work [17], we successfully adapted multilayer perceptron (MLP) classifiers trained with simulated annealing for AUC optimization. Now we aim to do so in a generalized manner so that, rather than developing new algorithms, our approach is to reuse the vast experience encoded into existing MLP training algorithms, devising a method through which they can be adapted with little effort, leaving untouched their core logic. This method is based on a novel AUC error metric (loss function) herewith defined that replaces the error metrics used in existing training algorithms to guide their training processes, therefore requiring simply the substitution of the error calculation routines of any MLP implementation.

This paper is structured as follows. Section 2 defines the novel AUC error metrics just mentioned and describes the method used to compute them efficiently. Section 3 explains the experimental setup devised to validate our approach by injecting the proposed metric into four existing MLP training algorithms and the results obtained. Finally, in Section 4 we draw the conclusions and outline future work.

II. AUC OPTIMIZATION IN MULTILAYER PERCEPTRON BASED CLASSIFIERS

This section formally defines the proposed method to adapt existing MLP training algorithms for AUC optimization in the sense just described. First, we settle for the notation defined in Table 1 to refer to the different elements in of a binary classification task (dataset elements, binary MLP classifiers and error measures). Then, we use the definition of the Mann-Whitney statistic for AUC [3] to obtain an AUC based error measure (loss function) and discuss its theoretical properties. Finally, since AUC calculation is typically expensive, we also provide an algorithm for an efficient error-bound approximation of the AUC, ensuring our method does not render modified MLP training algorithms impracticable.

A. Generalities

Since we are dealing with AUC optimization we assume the case of MLP based binary classifiers having two output

 ISBN: 978-1-61208-154-0

neurons: one positive neuron (representing the *positiveness* assigned by the MLP to the input vector) and one negative neuron (representing its *negativeness*), and we use the definitions detailed in Table 1.

TABLE 1: DEFINITIONS FOR BINARY MULTILAYER PERCEPTRONS

$\mathcal{X} \subset \mathbb{R}^p$	Domain of elements consisting of input vectors (with p features)
$\mathcal{Y} = \{P, N\}$	The two classes into which input vectors are classified
$(x, y), x \in \mathcal{X}, y \in \mathcal{Y}$	Element with its associated class (for supervised training)
$SD = \{(x_1, y_1), \ldots, (x_n, y_n)\}$	Dataset (for supervised training)
$S_P = \{x \mid (x, P) \in SD\}$	Positive elements of dataset
$S_N = \{x \mid (x, N) \in SD\}$	Negative elements of dataset
$S = S_P \cup S_N$	Elements of a dataset
$S(x) = y$	Class associated to element x through dataset S
$\lvert S \rvert = n$	Size of dataset
$\mathcal{F} = \{ f: \mathcal{X} \to \mathcal{Y} \}$	Set of functions representing binary classifiers
$h \in \mathcal{F}$	A binary classifier
$h(x) \in \mathcal{Y}$	Output of binary classifier h when applied to element x
$h_{sc}(x) \in \mathbb{R}$	Score assigned by binary classifier h to element x it is typically used by h to determine $h(x)$ by applying some threshold, and obtain ROC curves
$\mathcal{E}(S, h)$	A global error measure of classifier h when applied to training set S
$e(x, h)$	An individual error measure of classifier h when applied to element x
$Az(S, h)$	Area under the ROC curve (AUC) of dataset S when classified with h
$\mathcal{O} = [nn_{min}, nn_{max}] \subset \mathbb{R}$	Range of output values for the two output neurons of a binary MLP
$h_P(x), h_N(x) \in \mathcal{O}$	Output of the positive and negative neurons of the MLP based binary classifier h upon element x
$d_P(x), d_N(x) \in \mathcal{O}$	Ideal value for positive and negative neurons for x
$e_P(x, h), e_N(x, h)$	Error measures for the output of the positive and negative neurons of binary MLP h upon element x

At this point, given a binary classifier h and a dataset S, we distinguish two kinds of MLP training algorithms: (1) those that iterate through all dataset elements, using the error measures of each element $x \in S$ at the positive and negative output neurons, denoted by $e_P(x, h)$ and $e_N(x, h)$, and (2) those using the global error measure for the whole dataset, $\mathcal{E}(S, h)$, (see Table 1). We name the first kind of algorithms as *element error training algorithms* and the second kind as *dataset error training algorithms*. Notice that dataset error training algorithms only use $\mathcal{E}(S, h)$ regardless how it is calculated. Although typically a global error measure $\mathcal{E}(S, h)$ for a whole dataset is obtained by iterating over the error measures of all elements (such as by calculating their mean), this might not always be the case.

For a given dataset element $x \in S$ we define the ideal values at the positive and negative output neurons as:

$$\begin{aligned} d_P(x) &= nn_{\max} \quad d_N(x) = nn_{\min} \quad if \quad S(x) = P \\ d_P(x) &= nn_{\min} \quad d_N(x) = nn_{\max} \quad if \quad S(x) = N \end{aligned} \tag{1}$$

and fix a score metric, which linearly transforms the output of the two neurons to the [0,1] interval according to eq. 1, so that a score of 0.0 corresponds to an ideal negative element ($h_P(x) = nn_{min}$ and $h_N(x) = nn_{max}$) and a score of 1.0 corresponds to an ideal positive element ($h_P(x) = nn_{max}$ and $h_N(x) = nn_{min}$)

$$h_{sc}(x) = \frac{h_P(x) - h_N(x)}{2(nn_{max} - nn_{min})} + \frac{1}{2} \tag{2}$$

It can be easily proven that this definition ensures that $h_{sc}(x)$ stays within the [0,1] interval. In fact, for ROC purposes this restriction is not strictly needed as what matters is the relative ordering between positive and negative dataset elements induced by the score h_{sc} assigned to each one.

Commonly, a distance metric measures the error at the neuron's output with respect to the ideal output:

$$\begin{aligned} \Delta_P(x, h) &= d_P(x) - h_P(x) \\ \Delta_N(x, h) &= d_N(x) - h_N(x) \end{aligned} \tag{3}$$

B. Root Mean Square error measures

Based on the previous definitions, a Root Mean Square (RMS) error measure is commonly established for a dataset element and for the whole dataset as follows:

$$\begin{aligned} e^{RMS}(x, h) &= \sqrt{\frac{\Delta_P(x, h)^2 + \Delta_N(x, h)^2}{2}} \\ \mathcal{E}^{RMS}(S, h) &= \frac{\sum_{x \in S} e^{RMS}(x, h)}{|S|} \end{aligned} \tag{4}$$

Having, therefore, $\mathcal{E}^{RMS}(S, h)$ as the mean of $e^{RMS}(x, h)$. Then, $e^{RMS}(x, h)$ is simply mapped to each output neuron using the distance metric of eq. 3 as follows:

$$\begin{aligned} e_P^{RMS}(x, h) &= \Delta_P(x, h) \\ e_N^{RMS}(x, h) &= \Delta_N(x, h) \end{aligned} \tag{5}$$

In this case, dataset error training algorithms would use $\mathcal{E}^{RMS}(S, h)$, whereas element error training algorithms would use $e_P^{RMS}(x, h)$ and $e_N^{RMS}(x, h)$.

C. AUC Error Metrics

We want to make a AUC based error measure so that it can be injected back into training algorithms by either substituting $\mathcal{E}^{RMS}(S, h)$ or $e_P^{RMS}(x, h)$ and $e_N^{RMS}(x, h)$

 ISBN: 978-1-61208-154-0

without altering the rest of the logic. For this, we use the definition of the Mann-Whitney statistic for AUC [3]:

$$AUC(S,h) = \frac{\sum_{p\in S_P}\sum_{n\in S_N}\mathbf{1}[h_{sc}(n) < h_{sc}(p)]}{|S_P|\cdot|S_N|} \quad (6)$$

where $1[L]$ denotes the indicator function, yielding 1 if L is true and 0 otherwise. Through this, the contribution of dataset element x to $AUC(S,h)$ is established as:

$$AUC(x,h) = \begin{cases} \frac{\sum_{n\in S_N}\mathbf{1}[h_{sc}(n) < h_{sc}(x)]}{|S_P|\cdot|S_N|} & if\ x \in S_P \\ \frac{\sum_{p\in S_P}\mathbf{1}[h_{sc}(x) < h_{sc}(p)]}{|S_P|\cdot|S_N|} & if\ x \in S_N \end{cases} \quad (7)$$

Notice the fact that the maximum possible values for $AUC(h,x)$ are reached when the score of x is greater than the score of all negative elements if x is a positive element (and inversely when x is a negative element):

$$AUC_{MAX}(x,h) = \begin{cases} \frac{|S_N|}{|S_P|\cdot|S_N|} = \frac{1}{|S_P|} & if\ x \in S_P \\ \frac{|S_P|}{|S_P|\cdot|S_N|} = \frac{1}{|S_N|} & if\ x \in S_N \end{cases} \quad (8)$$

With this, we define the following error measures for dataset elements and for the whole dataset.

$$e^{AUC}(x,h) = 1 - \frac{AUC(x,h)}{AUC_{MAX}(x,h)}$$
$$\mathcal{E}^{AUC}(S,h) = 1 - AUC(S,h) \quad (9)$$

It would be tempting to define $e^{AUC}(x,h) = AUC_{MAX}(x,h) - AUC(x,h)$, however, $AUC_{MAX}(x,h)$ is usually a very small value (specially for large datasets), which would make it unpractical for MLP training purposes. In addition, this definition preserves the fact that the dataset error measure is the mean of the elements error measure, such as is the case between $\mathcal{E}^{RMS}(S,h)$ and $e^{RMS}(x,h)$.

Lemma 1: $\mathcal{E}^{AUC}(S,h)$ is the mean of $e^{AUC}(x,h)$ over the dataset elements as defined in eq. 9

Proof: Exploiting the fact that a binary dataset can be split into positive and negative elements:

$$\sum_{x\in S} e^{AUC}(x,h) = \sum_{p\in S_P} e^{AUC}(p,h) + \sum_{n\in S_N} e^{AUC}(n,h)$$

$$= \sum_{p\in S_P} 1 - \frac{AUC(p,h)}{AUC_{max}(p,h)} + \sum_{n\in S_N} 1 - \frac{AUC(n,h)}{AUC_{max}(n,h)}$$

$$= |S_P| - \sum_{p\in S_P}|S_P|\cdot AUC(p,h) + |S_N| - \sum_{n\in S_N}|S_N|\cdot AUC(n,h)$$

$$= |S_P| - \sum_{p\in S_P}|S_P|\cdot\frac{\sum_{n\in S_N}\mathbf{1}[h_{sc}(n) < h_{sc}(p)]}{|S_P|\cdot|S_N|} + |S_N| - \sum_{n\in S_N}|S_N|\cdot\frac{\sum_{p\in S_P}\mathbf{1}[h_{sc}(n) < h_{sc}(p)]}{|S_P|\cdot|S_N|}$$

$$= |S_P| - |S_P|\cdot\frac{\sum_{p\in S_P}\sum_{n\in S_N}\mathbf{1}[h_{sc}(n) < h_{sc}(p)]}{|S_P|\cdot|S_N|} + |S_N| - |S_N|\cdot\frac{\sum_{n\in S_N}\sum_{p\in S_P}\mathbf{1}[h_{sc}(n) < h_{sc}(p)]}{|S_P|\cdot|S_N|}$$

$$= |S_P| - |S_P|\cdot AUC(S,h) + |S_N| - |S_N|\cdot AUC(S,h)$$

$$= |S_P|\cdot\big(1 - AUC(S,h)\big) + |S_N|\cdot\big(1 - AUC(S,h)\big)$$

$$= (|S_P| + |S_N|)\cdot\big(1 - AUC(S,h)\big) = |S|\cdot\big(1 - AUC(S,h)\big)$$

$$= |S|\cdot\mathcal{E}^{AUC}(S,h)$$

$$\Rightarrow \mathcal{E}^{AUC}(S,h) = \frac{\sum_{x\in S} e^{AUC}(x,h)}{|S|}$$

■

However, $e^{AUC}(x,h)$ still needs to be mapped to each output neuron of a binary MLP classifier, which we do in the following way:

$$e_P^{AUC}(x,h) = e^{AUC}(x,h)\cdot\frac{\Delta_P(x,h)}{|\Delta_P(x,h)| + |\Delta_N(x,h)|}$$
$$e_N^{AUC}(x,h) = e^{AUC}(x,h)\cdot\frac{\Delta_N(x,h)}{|\Delta_P(x,h)| + |\Delta_N(x,h)|} \quad (10)$$

Therefore, $e^{AUC}(x,h)$ is distributed between the positive and negative neurons according to how far each one is from their ideal value, maintaining the direction of the distance metric (Δ_P and Δ_N). This way, using eq.10, dataset elements having a perfect AUC score, where $AUC(x,h)$ is maximum, do not produce any error even if the output values of the output neurons are not the ideal ones. In the limit case, where $|\Delta_P(x,h)| + |\Delta_N(x,h)| = 0$, we establish both $e_P^{AUC}(x,h) = 0$ and $e_N^{AUC}(x,h) = 0$.

With this, we inject the proposed $\mathcal{E}^{AUC}(S,h)$ error measure by replacing $\mathcal{E}^{RMS}(S,h)$ for dataset error training algorithms(such as simulated annealing and genetic algorithms as described below), whereas $e_P^{AUC}(x,h)$ and $e_N^{AUC}(x,h)$ replace $e_P^{RMS}(x,h)$ and $e_N^{RMS}(x,h)$ respectively forelement error training algorithms (such as backpropagation and resilient backpropagation as described below). Most importantly, in practical terms, using $\mathcal{E}^{AUC}$ and e^{AUC} in existing MLP training algorithms just amounts to substituting the error calculation routine without altering the rest of the algorithm logic.

 ISBN: 978-1-61208-154-0

D. Efficient AUC Calculation

One drawback of using the proposed error measures is that AUC calculation is computationally expensive, since it usually requires full sorting of the measured dataset. This is specially relevant when using AUC based metrics in iterative algorithms, such as in MLP, since it may render good theoretical or experimental results impractical to use. We observed that commonly used AUC calculation techniques, such as the one provided by Weka [18], slow down about 5 times the MLP training algorithms described in next section and modified to use $\mathcal{E}^{AUC}(S,h)$ and $e^{AUC}(x,h)$ as just explained. To overcome this, we developed an efficient AUC calculation method that approximates the actual value to an arbitrary maximum error established by the user. It is based on discretizing the score space for each dataset element and, therefore, removing the need to full sorting. It produces all necessary metrics described in previous section namely, $AUC(x,h)$ and $e^{AUC}(x,h)$ for each dataset element and, of course, $AUC(S,h)$ and $\mathcal{E}^{AUC}(S,h)$ for the whole dataset.

Recalling that $h_{sc}(x) \in [0,1]$ for all elements x of a dataset, our AUC error bounded approximation method is based on the observation that, to compute the contribution of each positive element to the dataset AUC, $AUC(x,h), x \in S_P$, we are interested only on the number of negative elements whose score is lower to the score of x, regardless their actual rank. So somehow, full sorting is not totally necessary. Intuitively, our method splits the [0,1] interval into contiguous non-overlapping subintervals of equal length and counts the number of positive and negative elements whose score falls within each subinterval. This operation requires one dataset scan and elementary arithmetic. Then, for each positive element $p \in S_P$, the expression $1[h_{sc}(n) < h_{sc}(p)]$ in eq. 6 or eq. 7 is approximated by counting the number of negative elements of the subintervals under the subinterval to which p belongs. Only the negative elements falling within the same interval as p will not be counted and constitute the source of the approximation error. The finer the [0,1] interval split, the more accurate the approximation will be. This process can be iterated until the approximation error falls below a user defined value involving only dataset elements falling within intervals producing the greatest errors. With the datasets used in this work, experiments showed that with two or three further iterations the approximation error always fell under 0.001, which is good enough to ensure dispensable influence in MLP accuracy. With this approximation, MLP training algorithms were slowed down only 1.5 times in average, when comparing the RMS error based original algorithms with the AUC error modified ones.

III. EXPERIMENTAL VALIDATION

A set of experiments was set up to measure the behavior of the proposed method for AUC optimization upon existing MLP training algorithms, which were modified to use the definitions described in previous section. Experiments were carried out by using selected datasets from the UCI machine learning repository and the goal was to compare the AUC performance of the original MLP training algorithms (aiming at minimizing the error rate) against their modified versions through the same training conditions.

A. MLP Training Algorithms

Four different MLP training algorithms were used as implemented within the Encog toolkit [19], which use RMS error metrics, and modified them to use the AUC error metrics defined in previous section:

Feed Forward Back Propagation (FFBP): The classical *element error training algorithm*, where per-element error measures at each output neuron, $e_P(x,h)$ and $e_N(x,h)$, are used to adjust neuron weights of the various layers of the MLP backwards from the output layer to the input layer, through a gradient descent method controlled by two user definable parameters: the *learning rate* and the *momentum.*

Feed Forward Resilient Propagation (FFRP): Also an *element error training algorithm*, since it is a variation of FFBP where each neuron has its own set of independent parameters to control the gradient descent (similar to the FFBP learning rate and momentum) that the algorithm adjusts automatically throughout the training process.

Feed Forward Simulated Annealing (FFSA): A *dataset error training algorithm*, where an MLP is taken through several "cooling" cycles. Starting at an initial top temperature, at each step in each cooling cycle the MLP weights are randomized according to the temperature (higher temperatures produce higher random variability) generating a new MLP. If the new MLP produces a lower error on the whole dataset, $\mathcal{E}(S,h)$, it is kept to the next cooling step. Otherwise it is discarded. Then, the temperature is lowered one step and the process continues. The user definable parameters it accepts are *start-temperature*, *end-temperature* and *number-of-cycles.*

Feed Forward Genetic Algorithms (FFGA): A dataset error training algorithm, where the vector of MLP weights is interpreted as a chromosome and a population of MLPs with identical structure and different weights is evolved through generations that mate and cross over. MLPs (chromosomes) yielding lower errors on the whole dataset, $\mathcal{E}(\mathrm{S},\mathrm{h})$, are considered as best suited and, therefore, with a higher probability of survival and mating to the next generation. The user definable parameters it accepts are *population-size*, *mutation-percent* and *percent-to-mate-with*.

For a given algorithm specific values of its required training parameters constitute a training configuration. We used the same training configuration in the original and the modified algorithms to facilitate comparisons. The term "original algorithms" is therefore used for FFBP, FFRP, FFGA and FFSA as originally delivered by Encog (using RMS based error measures), and the term "modified algorithms" is used for their counterparts (FFBPROC, FFRPROC, FFGAROC and FFSAROC) adapted by the authors to use the previously defined AUC based error measures.

Since many different MLP configurations were to be evaluated for different training algorithms and datasets, a software framework (named BiomedTK [20]) was specially developed to manage their evaluation over distributed

 ISBN: 978-1-61208-154-0

computer clusters. BiomedTK stands for Biomedical Data Analysis Toolkit, and implements different kinds of dataset normalizations, validation procedures (n fold cross-validation, leave one out, etc.), distribution of configuration into jobs, management of jobs and training results, etc. This allowed us to efficiently harness computing power to evaluate a complex variety of training algorithms, datasets and training configurations by using just a few configuration artifacts, such as the exploration definition file shown in Table 3.

B. Selected Datasets and MLP configurations

Table 2 shows the datasets selected for experimentation from the UCI repository. The criteria to select those datasets was: (1) they are binary datasets, (2) they provide a diversity of skews in their class distribution, (3) they represent classification tasks of different nature and (4) they contain less than 1000 elements, which makes MLP training affordable from a computational point of view. Class skew was considered important since AUC is known to be insensitive to class distribution.

TABLE 2: BINARY UCI DATASETS USED FOR VALIDATION

dataset	elements	features	class skew
pgene	106	57	50%
E. Coli promoter gene sequences (DNA) with partial domain theory			
mmass	961	5	54%
Discrimination of benign and malignant mammographic masses based on BI-RADS attributes and the patient's age.			
heartsl	270	13	56%
Heart disease database reformatted			
liver	345	6	58%
BUPA Medical Research Ltd. database on liver desease			
bcwd	569	30	63%
Diagnostic breast cancer Wisconsin database.			
pimadiab	768	8	65%
From National Institute of Diabetes, Digestive and Kidney Diseases; Includes cost data.			
tictac	958	9	65%
Possible configurations of tic-tac-toe game.			
echocard	131	8	67%
Patients surviving for at least one year after a heart attack.			
haber	306	3	74%
Dataset contains cases from study conducted on the survival of patients who had undergone surgery for breast cancer.			
park	195	22	75%
Oxford Parkinson's Disease Detection Dataset.			
glass	214	9	76%
From USA Forensic Science Service; 6 types of glass; defined in terms of their oxide content (i.e. Na, Fe, K, etc.).			
spectf	267	44	79%
Data on cardiac Single Proton Emission Computed Tomography (SPECT) images.			

For each dataset, a set of MLP configurations was defined for each original algorithm and its modified version (FFSA/FFSAROC, FFGA/FFGAROC, FFBP/FBPROC, FFRP/FFRPROC). Table 3 shows an example configuration file used through BiomedTK to manage the exploration of training configurations for FFSA and FFSAROC MLPs with the SPECTF dataset, which results in 24 configurations for FFSA and another 24 configurations for FFSAROC. Configurations include MLPs with one or two hidden layers, with 89 or 178 neurons in the first hidden layer, with the parameter start-temperature set to 30 or 100, etc.

Similar configurations sets were defined for each algorithm and dataset, fixing the particular parameters of each training algorithm to be the same for all datasets and varying only the number of input neurons according to the dataset input features, while keeping the same proportions in the number of neurons of the hidden layers with respect to the input layer (as in the example in Table 3).

TABLE 3: FFSA/FFSAROC EXPLORATIONS FOR SPECTF DATASET.

```
explore.neurons.input                = 44
explore.neurons.output               = 2
explore.neurons.layer.01             = 89:178
explore.neurons.layer.02             = 44:132
explore.activation.function          = sigm
explore.trainingsets                 = spectf
explore.trainengines                 = encog.ffsa:encog.ffsaroc
explore.validation                   = cvfolds 10
explore.encog.ffsa.starttemp         = 30:100
explore.encog.ffsa.endtemp           = 2:10
explore.encog.ffsa.cycles            = 50
explore.encog.ffsa.stop.epochs       = 3000
explore.encog.ffsa.stop.error        = 0.0001
explore.encog.ffsaroc.starttemp      = 30:100
explore.encog.ffsaroc.endtemp        = 2:10
explore.encog.ffsaroc.cycles         = 50
explore.encog.ffsaroc.stop.epochs    = 3000
explore.encog.ffsaroc.stop.error     = 0.0001
explore.numberofjobs                 = 40
```

Each MLP configuration was trained with 10-fold cross-validation. In total, 180 MLP configurations were trained for each of the 12 datasets, 90 MLP configurations corresponding to the original algorithms and 90 to their corresponding modified versions. Overall, 2160 MLP configurations were trained on a cluster with 50 computers, which, dedicated, took about 4 full physical days. Evaluation of all these configurations over the computing resources was managed through the BiomedTK software framework as mentioned above.

C. Results

For each dataset and training algorithms, results were processed and averaged in the following way: (1) each MLP configuration was trained both with the original algorithm from Encog and its modified version; (2) AUC results from all configurations trained with the original algorithms is averaged and its standard deviation is calculated and (3) AUC results from all configurations trained with the modified algorithms is averaged and its standard deviation is calculated. The percentage of improvement of the averages (positive or negative) obtained by the modified version was calculated with respect to the original version through the following formula:

 ISBN: 978-1-61208-154-0

$$improv = 100 \times \frac{MODIFIED_{avg} - ORIGINAL_{avg}}{ORIGINAL_{avg}} \quad (11)$$

Table 4 summarizes the average AUC obtained per dataset and category of algorithm (dataset error based or element error based), aggregating them in total (column 'OVERALL'). Finally, Table 5 provides some correlation measures between different obtained measures.

TABLE 4: AUC IMPROVEMENT PER DATASET AND ALGORITHM TYPE

	OVERALL AUC			**ELMENT BASED AUC (FFBP and FFRP)**			**DATASET BASED AUC (FFSA and FFGA)**		
	orig	*mod*	*impro*	*orig*	*mod*	*impro*	*orig*	*mod*	*impro*
pgene	0,742	0,731	-1,01%	0,786	0,765	-2,44%	0,698	0,698	0,43%
mmass	0,781	0,827	6,65%	0,708	0,782	11,23%	0,855	0,872	2,07%
heartsl	0,795	0,854	7,93%	0,740	0,813	10,37%	0,849	0,894	5,48%
liver	0,622	0,707	14,59%	0,605	0,684	13,91%	0,640	0,731	15,26%
bcwd	0,900	0,903	0,20%	0,831	0,826	-0,83%	0,969	0,980	1,23%
pimadiab	0,704	0,741	5,19%	0,635	0,665	4,32%	0,774	0,818	6,07%
tictac	0,733	0,786	7,81%	0,747	0,794	6,87%	0,720	0,778	8,75%
echocard	0,566	0,623	11,01%	0,507	0,605	19,45%	0,626	0,641	2,57%
haber	0,629	0,674	7,30%	0,580	0,635	9,33%	0,678	0,714	5,28%
park	0,840	0,846	1,18%	0,800	0,804	1,08%	0,879	0,889	1,27%
glass	0,876	0,899	3,15%	0,795	0,838	5,85%	0,956	0,960	0,46%
spectf	0,673	0,712	6,25%	0,616	0,660	7,47%	0,730	0,764	5,02%
averages	0,738	0,775	5,86%	0,696	0,739	7,22%	0,781	0,812	4,49%

As it can be observed, there is a generalized AUC improvement by our proposed method, having occasional degradations in particular datasets (mostly in pgene and bcwd). The global averaged improvement is 5.86% with a great variability on each dataset and algorithm. It can also be observed that improvement is greater in element error training algorithms (FFBP and FFRP) than in dataset error training algorithms, although this might be due to the fact that these later ones tend to give better results as shown by the correlation between improvement and ORIGINAL AUC avg in Table 5 line 2 (improvement is greater when ORIGINAL AUC is lower). We acknowledge that this last observation might be biased by the way *improv* is defined (eq. 11) since greater AUC leave less room for improvement.

TABLE 5: CORRELATIONS BETWEEN EXPERIMENT MEASURE PAIRS

	FFBP	FFRP	FFSA	FFGA	ALL
improvement vs. class skew	-0,257	0,259	0,103	-0,208	-0,026
improvement vs. ORIGINAL AUC avg	-0,365	-0,872	-0,664	-0,553	-0,717
improvement vs. ORIGINAL AUC stddev	0,063	0,454	0,362	0,208	0,178
ORIGINAL AUC stddev vs. MODIFIED AUC stddev	0,883	0,631	0,884	0,828	0,682
ORIGINAL AUC avg vs. MODIFIED AUC avg	0,848	0,956	0,981	0,956	0,971

Other interesting observations are the following:

A) Both AUC averages and standard deviations are strongly correlated between the original algorithms and their modified versions (Table 5 lines 4 and 5).

B) Small standard deviations result from MLP configurations producing similar AUC scores (all configurations classify the dataset as good or as bad), whereas larger standard deviations result from some MLP configurations producing significantly better AUC scores than others. The strong correlations observed in averages and standard deviations leads to think that modified algorithms behave similarly to the original ones in the sense that they respond in the same way to dataset particularities (difficulty or easiness to classify).

C) Class skew seems to have little influence on improvement (Table 5 line 1), or at least in a non-homogeneous way across the different training algorithms.

D) Except in the case of FFBP, there seems to be a significant correlation (Table 5 line 3) between the standard deviation of the ORIGINAL AUC and the improvement obtained by our method in the positive direction (increasing standard deviation with increasing improvement). Large standard deviations may occur in many scenarios, such as when a dataset is hard to separate and well performing MLP configurations are scarce. The observed correlation might suggest that our method could be more appropriate in these situations to increase overall MLP AUC performance.

All these issues might be subject of further research, seeking stronger statistical evidence to support the hypothesis outlined.

IV. CONCLUSIONS AND FUTURE WORK

This work presented a new method to insert AUC based error metrics in existing MLP training algorithms for AUC optimization. In practical terms, the proposed approach only requires the substitution of the error computing routines of the underlying training algorithms, respecting their core logic. Experimental evidence demonstrated a consistent improvement in AUC through a variety of datasets and training algorithms requiring little coding effort. In addition, and equally important, an efficient method has been develop providing an error bound approximation for the AUC, which ensures MLP training remains computationally affordable. Finally, we can conclude that the newly developed AUC error metrics show a consistent behavior in both its theoretical definition and experimental results.

Future work intends to further validate this approach in other kinds of machine learning algorithms and explore its theoretical implications, specially in those algorithms requiring continuity conditions on the loss or error functions (such as gradient descent). Also stronger statistical evidence for the hypotheses outlined in previous section will be pursued.

ACKNOWLEDGMENTS

This work is part of the GRIDMED research collaboration project between INEGI (Portugal) and CETA-CIEMAT (Spain). Prof. Guevara acknowledges POPH - QREN-Tipologia 4.2 – Promotion of scientific employment funded by the ESF and MCTES, Portugal. CETA-CIEMAT acknowledges the support of the European Regional Development Fund.

 ISBN: 978-1-61208-154-0

REFERENCES

[1] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognition Letters,* vol. 27, pp. 861-874, 2006.

[2] M. Zweig and G. Campbell, "Receiver-operating characteristic (ROC) plots: a fundamental evaluation tool in clinical medicine " *Clinical Chemistry,* vol. 39, pp. 561-577, 1993.

[3] J. A. Hanley and B. J. McNeil, "The meaning and use of the area under a receiver operating characteristic (ROC) curve," *Radiology,* vol. 143, pp. 29-36, 1982.

[4] J. Iavindrasana*, et al.*, "Clinical data mining: a review," *Yearb Med Inform,* pp. 121-33, 2009 2009.

[5] T. Fawcett, "ROC Graphs: Notes and Practical Considerations for Researchers," HP Labs Tech Report HPL-2003-4, 2003.

[6] A. P. Bradley, "The use of the area under the ROC curve in the evaluation of machine learning algorithms," *Pattern Recognition,* vol. 30, pp. 1145-1159, 1997.

[7] C. Cortes and M. Mohri, "AUC optimization vs. error rate minimization," *Advances in Neural Information Processing Systems 16,* vol. 16, pp. 313-320, 2004.

[8] U. Brefeld and T. Scheffer, "AUC Maximizing Support Vector Learning," *Proceedings of the ICML Workshop on ROC Analysis in Machine Learning,* 2005.

[9] T. Calders and S. Jaroszewicz, "Efficient AUC optimization for classification," *Knowledge Discovery in Databases: PKDD 2007, Proceedings,* vol. 4702, pp. 42-53, 2007.

[10] L. Costa*, et al.*, "Tuning Parameters of Evolutionary Algorithms Using ROC Analysis," in *2nd International Workshop on Practical Applications of Computational Biology and Bioinformatics (IWPACBB 2008)*. vol. 49, J. Corchado*, et al.*, Eds., ed: Springer Berlin / Heidelberg, 2009, pp. 217-222.

[11] F. Provost and T. Fawcett, "Robust Classification for Imprecise Environments," *Machine Learning,* vol. 42, pp. 203-231, 2001.

[12] C. Marrocco*, et al.*, "Maximizing the area under the ROC curve by pairwise feature combination," *Pattern Recogn.,* vol. 41, pp. 1961-1974, 2008.

[13] C. L. Castro and A. P. Braga, "Optimization of the Area under the ROC Curve," in *Neural Networks, 2008. SBRN '08. 10th Brazilian Symposium on*, 2008, pp. 141-146.

[14] K. A. Toh*, et al.*, "Maximizing area under ROC curve for biometric scores fusion," *Pattern Recognition,* vol. 41, pp. 3373-3392, Nov 2008.

[15] A. K. S. Wong*, et al.*, "Improving text classifier performance based on AUC," *18th International Conference on Pattern Recognition, Vol 3, Proceedings,* pp. 268-271, 2006.

[16] G. Han and C. Zhao, "AUC maximization linear classifier based on active learning and its application," *Neurocomputing,* vol. 73, pp. 1272-1280, 2010.

[17] R. Ramos-Pollan*, et al.*, "Introducing ROC curves as error measure functions. A new approach to train ANN-based biomedical data classifiers," in *15th Iberoamerican Congress on Pattern Recognition*, Sao Paolo, Brasil, 2010.

[18] Mark Hall*, et al.*, "The WEKA Data Mining Software: An Update," *SIGKDD Explorations,* vol. 11, 2009.

[19] J. Heaton, "Programming Neural Networks with Encog 2 in Java," ed: Heaton Research, Inc., 2010.

[20] R. Ramos-Pollán et al., "A Software Framework for Building Biomedical Machine Learning Classifiers through Grid Computing Resources," *Journal of Medical Systems*, 1-13, DOI 10.1007/s10916-011-9692-3

 ISBN: 978-1-61208-154-0

Schrödinger's Register: Foundational Issues and Physical Realization

Stephen Pink
School of Computing and Communication
Lancaster University
Lancaster, UK
pink@comp.lancs.ac.uk

Stanley Martens
School of Accountancy and Management Information Systems
DePaul University (retired)
Chicago, IL USA
smartens00@gmail.com

***Abstract*—This work-in-progress paper consists of four points which relate to the foundations and physical realization of quantum computing. The first point is that the qubit cannot be taken as the basic unit for quantum computing, because not every superposition of bit-strings of length n can be factored into a string of n-qubits. The second point is that the "No-cloning" theorem does not apply to the copying of one quantum register into another register, because the mathematical representation of this copying is the identity operator, which is manifestly linear. The third point is that quantum parallelism is not destroyed only by *environmental decoherence*. There are two other forms of decoherence, which we call *measurement decoherence* and *internal decoherence*, that can also destroy quantum parallelism. The fourth point is that processing the contents of a quantum register "one qubit at a time" destroys entanglement.**

***Keywords*-qubit; entanglement; decoherence; no-cloning theorem; quantum register.**

I. INTRODUCTION

This paper will make four points. Points (A) and (B) are foundational. Points (B), (C), and (D) relate to the physical realization of quantum computing. We will state the points and then elaborate on them.

A. The basic element of quantum computing is not the *qubit* but the *q-string.* The qubit is not basic because not every q-string can be factored into a string of qubits.

B. A *processing step* in quantum computing is defined as the application of a unitary linear operator on q-strings [4]. For physical realization purposes, this definition is incomplete. In a real quantum computer, q-strings of length n "live" on n-bit registers in superposed states. To specify a processing step, one must specify the input and output registers. Let ψ be a q-string. A "cloning" function f_{12} can be defined as $f(\psi$ on register $1) = \psi$ on register 2; the function copies ψ from register 1 to register 2. The existence of such a processing step does not violate the "No-cloning" theorem.

C. The power of quantum computing depends on quantum parallelism. Quantum parallelism is destroyed if q-strings decohere during processing. Experimental results indicate that the time to environmental decoherence is inversely related to the size of the physical system considered. If this is so, then the longer the q-string, the shorter the time to its environmental decoherence. This rules out quantum parallelism for q-strings of arbitrary length. Besides environmental decoherence, two other kinds of decoherence are introduced and discussed.

D. If a q-string is processed "one qubit at a time", then the resulting q-string is a string of qubits. So, any entanglement in the original q-string is destroyed.

II. ELABORATIONS OF THE FOUR POINTS

A. Qubits and Q-Strings

A *bit* = b_i is 0 or 1. A string of bits of length $n = \mid b_1 \ldots b_n\rangle$. The number of all strings of bits of length $n = 2^n$. A *q-string of length n* is a sum of the bit strings of length *n* weighted by complex numbers. So, a q-string of length $n =$

$$\psi = \sum_{m=1}^{m=2^n} c_m \phi_m,$$

where the ϕ_m are the bit strings of length n and the complex numbers c_m satisfy the condition $\sum_m \mid c_m \mid^2 = 1$. Some of the c_m may be 0, so it may be that not every bit string of length n is a nonzero-weighted component of the q-string. A *qubit* q_i = a q-string of length 1. So, $q_i = \alpha_i \mid 0\rangle + \beta_i \mid 1\rangle$, where $\mid \alpha_i \mid^2 + \mid \beta_i \mid^2 = 1$. A *string of qubits* is a product of qubits $= q_1 \otimes q_2 \otimes \ldots q_n$.

Consider a string of two qubits:

$$\begin{aligned} q_1 \quad \otimes \quad q_2 &= (\alpha_1 \mid 0\rangle + \beta_1 \mid 1\rangle) \otimes (\alpha_2 \mid 0\rangle + \beta_2 \mid 1\rangle) \\ = \quad & \alpha_1\alpha_2 \mid 00\rangle + \alpha_1\beta_2 \mid 01\rangle + \beta_1\alpha_2 \mid 10\rangle + \beta_1\beta_2 \mid 11\rangle. \end{aligned}$$

$\mid \alpha_1 \mid^2 + \mid \beta_1 \mid^2 = 1$ and $\mid \alpha_2 \mid^2 + \mid \beta_2 \mid^2 = 1$.

Now consider the q-string of length 2:

$$0 \mid 00\rangle \; + \; \frac{1}{\sqrt{2}} \mid 01\rangle \; + \; \frac{1}{\sqrt{2}} \mid 10\rangle \; + \; 0 \mid 11\rangle$$

 ISBN: 978-1-61208-154-0

If this q-string = $q_1 \otimes q_2$, then $\alpha_1\alpha_2 = 0$, $\alpha_1\beta_2 = \frac{1}{\sqrt{2}}$, $\beta_1\alpha_2 = \frac{1}{\sqrt{2}}$, and $\beta_1\beta_2 = 0$. So, either $\alpha_1 = 0$ or $\alpha_2 = 0$. But if $\alpha_1 = 0$, then $\alpha_1\beta_2 = 0 \neq \frac{1}{\sqrt{2}}$. If $\alpha_2 = 0$, then $\beta_1\alpha_2 = 0 \neq \frac{1}{\sqrt{2}}$. So the above q-string of length 2 is not a string of 2 qubits. This argument can be generalized. So, not every q-string of length n is a string of n qubits. The above q-string of length 2 is said to have "entangled qubits", because it cannot be factored into a string of 2 qubits.

B. Q-Strings and the No-cloning Theorem

The physical realization of a q-string of length n in a real quantum computer will be the state of a register with n bit positions. Suppose a real quantum computer contains two n-bit registers. Suppose register 1 contains a q-string of length n, i.e., suppose register 1 is in a superposition of states where each of the states is a bit-string of length n. Suppose register 2 contains a string of n zeroes. Nothing in what has been described so far rules out that the computer can execute the command: *Store the content of register 1 in register 2*. The result of this processing will be two registers, each containing the same q-string. Wouldn't this violate the "No-cloning theorem" [2]?

A *processing step* in quantum computing is defined as the application of a unitary linear operator f on q-strings of length n [4]. Suppose

$$\psi = \sum_{m=1}^{m=2^n} c_m\phi_m,$$

a q-string of length *n*. Since f is linear, $f(\psi) = \sum_{m=1}^{m=2^n} c_m f(\phi_m)$. Since f is unitary, the action of f on a bit string yields a bit string of the same length. So,

$$f(\phi_m) = \phi_{m\prime} = \phi_{f(m)}.$$

Thus,

$$f(\psi) = \sum_{m=1}^{m=2^n} c_m\phi_{f(m)}.$$

As far as it goes, this definition of *processing step* is correct. For physical realization purposes, however, it is incomplete. In a real quantum computer, quantum strings live on n-bit registers. So, the mathematical representation of a processing step must specify the input and output registers. Thus, the "cloning" function C must be identified as C_{12}. So, $C(\psi$ on register $1) = \psi$ on register 2, and C is simply the Identity Transformation, $I_\psi = \psi$, which is unitary and linear.

The "No-cloning theorem" does not apply here. What the No-cloning theorem states is as follows: Let ψ be a q-string of length n. Let 0 be a string of n zeroes. No-cloning result: there is no unitary linear function g on q-strings of length $2n$, such that $g(\psi \otimes 0) = g(\psi \otimes \psi)$.

C. Quantum Parallelism and Decoherence

What is *quantum parallelism*? Suppose ψ is a q-string and ψ has m different bit strings appearing as nonzero-weighted components. Then, quantum parallelism is the idea that one processing step on ψ is, in a sense, equivalent to m processing steps on the bit string components of ψ [2]. Since processing takes time, quantum parallelism is lost if the q-string *decoheres* during processing. What is *decoherence*? The only kind of decoherence discussed as such in quantum computing is *environmental* decoherence. We believe that there are two other forms of decoherence, *measurement* decoherence and *internal* decoherence, and that these other forms may pose obstacles for quantum parallelism as well. Let us start with environmental decoherence. Let

$$\psi = \sum_m c_m\phi_m$$

be the state of a physical system where the ϕ_m are the base states of the system and the c_m are the complex numbers satisfying the usual condition. Let E_0 represent the initial state of the environment. Environmental decoherence is the idea that after a time (decoherence time) the physical system interacts enough with the environment so that the state of the system plus environment evolves to the following:

$$\mid \psi, E_0\rangle \longrightarrow \sum_m c_m \mid \phi_m, E_m\rangle,$$

where the E_m are states of the environment that do not mutually "interfere". What the "non-interference" means practically is that the evolved state immediately collapses:

$$\sum_m c_m \mid \phi_m, E_m\rangle \longrightarrow \text{ one of the } \mid \phi_m, E_m\rangle \text{ states}$$

with a probability of $\mid c_m \mid^2$. Q-strings live on the register of the quantum computer. So, Ψ above is the state of the register(s) of the computer, and E above is the state of the environment of the register(s); i.e., the rest of the computer plus the external world. So, if decoherence time is less than processing time, a q-string will collapse into one of its component bit strings, and quantum parallelism will be destroyed. Erich Joos [1] states that experimental results seem to indicate that decoherence time is related inversely to size; he even says (p. 13): "..macroscopic objects are extremely sensitive and immediately decohered." If what Joos says is true, then the longer the q-string, the shorter the time to its decoherence. This rules out quantum parallelism for q-strings of arbitrary length. Joos says (p. 14):

> ...(decoherence) represents a major obstacle for people trying to construct a quantum computer. Building a really big one may well turn out to be as difficult as detecting other Everett worlds!

Many think that detecting other Everett worlds is impossible [3]. *Measurement* decoherence can be explained as follows.

Let

$$\psi = \sum_m c_m \phi_m$$

be the state of a physical system, and suppose at t_0 (the initial time), ψ is coupled with a *measuring device* in state M_0. Let "measurement time" be the amount of time required for the measuring device to measure the physical system, i.e., the amount of time for the measuring device to evolve from M_0 to a superposition of *indicator* states M_i. The picture of the evolution is as follows:

$$\sum_m c_m \phi_m M_0 \quad \longrightarrow \quad \sum_m c_m \phi_m M_m.$$

If we make the assumption that the $M_i (i \geq 1)$ do not mutually interfere, then ψ immediately collapses:

$$\sum_m c_m \phi_m M_m \quad \longrightarrow$$

$$\text{one of the } \phi_m M_m \text{ with a probability of } \mid c_m \mid^2 .$$

Measurement decoherence (called *quantum measurement* by quantum computer scientists) is a resource of, and not an obstacle to, quantum computing if it occurs *after processing is complete*. Measuring the output q-string is the way to read information contained in that q-string. No one will intentionally apply a measuring device to a register or registers *before* processing is complete. So, how can measurement decoherence be an obstacle to quantum computing? A physical quantum computer will contain a register or registers, but will also contain other devices (for processing, etc.) besides registers. If the "innards" of a physical quantum computer exclusive of the registers *act like* a measuring device during processing, then there will be an unintentional measurement of a register or registers during processing, and quantum parallelism will be lost. Thus, it is a challenge not only to build registers that can exist in superposed states, but also to build the rest of the quantum computer so that it does not act like a measuring device on registers during processing.

The third form of decoherence is *internal* decoherence. Suppose we have a physical system in an initial state ψ_0. Suppose also that in some time interval (evolution time), the physical system evolves to a superposition of base states:

$$\psi \quad \longrightarrow \quad \sum_m c_m \phi_m.$$

If we assume that the ϕ_m do not mutually interfere, we have immediate collapse:

$$\sum_m c_m \phi_m \longrightarrow \text{one of the } \phi_m \text{with a probability of } \mid c_m \mid^2 .$$

In the standard two-slit experiment, we have evolution to:

$$\alpha \mid \text{particle travels through slit 1}\rangle$$
$$+ \beta \mid \text{particle travels through slit 2}\rangle,$$

but these two states mutually interfere, as is evidenced by the interference pattern built up on the photographic backstop as the experiment is repeated. So the standard two-slit experiment is *not* an example of internal decoherence.

We can get internal decoherence if we modify the two-slit experiment. Put a light source near slit 1, so that a particle traveling through slit 1 produces a light flash because a photon from the source bounces off the particle. Then we have evolution to:

$$\alpha \mid \text{particle travels through slit 1} + \text{flash of light}\rangle$$
$$+ \beta \mid \text{particle travels through slit 2} + \text{no flash of light}\rangle.$$

These two states do not mutually interfere, as is evidenced by the lack of interference pattern on the photographic backstop. (Remember that *observation* of the light flash is not necessary to destroy interference; only existence of the flash is necessary.

Another physical system that internally decoheres is Schrödinger's Cat Box, consisting of a box occupied by a radioactive source, Geiger counter, trip hammer, vial of cyanide, and live cat. The box evolves into:

$$\alpha \mid \text{dead cat, smashed vial, tripped hammer, etc.}\rangle$$
$$+ \ \beta \mid \text{live cat, unsmashed vial, untripped hammer, etc.}\rangle$$

Based on all available observational evidence, these two states do not mutually interfere. No one has ever observed a superposition of $\alpha \mid \text{dead cat}\rangle \ + \ \beta \mid \text{live cat}\rangle$, let alone:

$$\alpha \mid \text{smashed vial}\rangle \ + \ \beta \mid \text{unsmashed vial}\rangle,$$
$$\alpha \mid \text{tripped hammer}\rangle \ + \ \beta \mid \text{untripped hammer}\rangle, \text{ etc.}$$

So, the solution to Schrödinger's Paradox is internal decoherence. We can talk of Schrödinger's Register instead of Schrödinger's Cat, and mean by this an n-bit register that can exist in a superposition of bit strings of length n such that those bit strings *do* interfere. (We want the bit strings to interfere, or else we would have a collapse to a single bit string and no quantum parallelism.) So, the challenge for quantum computer scientists is to build Schrödinger's Register. Good luck!

D. Qubits, Q-Strings, and Entanglement

Consider a q-string of length n, ψ. Suppose ψ is "entangled." Then it is not equal to a string of n qubits, but a string of n qubits can be constructed from it in the following way:

Survey the bit string components of ψ. Let m be a position from 1 to n in the bit string. Add the amplitudes for all components with a 0 in position m. Call the sum α_m. Add the amplitudes for all components with a 1 in position m. Call the sum β_m. Construct the qubit $(\alpha_m \mid 0\rangle + \beta_m \mid 1\rangle)$. Take the product of such qubits for all positions. This is the

 ISBN: 978-1-61208-154-0

string of qubits to be constructed:

$$\bigotimes_{m=1}^{m=n} (\alpha_m \mid 0\rangle \; + \; \beta_m \mid 1\rangle) \; = \; \bigotimes_{m=1}^{m=n} q_m \; .$$

The result of processing ψ "one qubit at a time" $=$ the product of the results of applying a processing step f on qubits to each qubit in the string constructed from (but not identified with) ψ:

$$\bigotimes_{m=1}^{m=n} f(q_m) \; = \; \bigotimes_{m=1}^{m=n} q'_m \; .$$

A string of m qubits has no entanglement among qubits. So, processing ψ "one qubit at a time" destroys entanglement.

III. Conclusion

We believe that the four points above will all be necessary to progress in understanding how to realize a quantum computer. Most fundamental would be the shift from qubit to q-string as the basic entity of quantum computation. At the same time it is important to see that "No Cloning" is not necessarily an obstacle to the moving of data between one register and another in a quantum computer, a necessary to realizing practical quantum computation. Also, we show that there are a number of types of decoherence, and that understanding the difference between these allows us to build a quantum register whose processes are relatively stable in the external environment of a real physical machine.

In our future research, we plan to explore more deeply the properties of a q-string versus a string of qubits and to present a more formal proof of the difference between these two concepts. We shall study the notion of entanglement and how it relates to quantum parallelism. In particular, since decoherence is the major obstacle to achieving quantum parallelism, we would like to understand better the relationship between entanglement and decoherence. In addition, we want to make clear how "No Cloning" affects the design and implementation of quantum memory and storage.

References

[1] E. Joos, *Elements of Environmental Decoherence*, (1999), e-print available at http://xxx.lanl.gov/abs/quant-ph/9908008v1.

[2] J. Gruska, *Quantum Computing Challenges*, In *Mathematics Unlimited, 2001 and Beyond*, Vol. 1. Berlin-Heidelberg : Springer-Verlag, 2000. ISBN 3-540-66913-2, pp. 529-563.

[3] D. Albert and B. Loewer *Interpreting the Many Worlds Interpretation*. SYNTHESE, Volume 77, Number 2, 195-213, DOI: 10.1007/BF00869434.

[4] J. Stolze and D. Suter, *Quantum Computing*, Wiley-VCH Verlag GMBH & Co., 2nd Edition, 2008, ISBN 978-3-527-40787-3, pp.6-7.

 ISBN: 978-1-61208-154-0

A Real-Time PC Based Software Radio DVB-T Receiver

Shu-Ming Tseng
Department of Electronic Engineering
National Taipei University of Technology
Taipei, Taiwan
shuming@ntut.edu.tw

Jian-Cheng Yu
Department of Electronic Engineering
National Taipei University of Technology
Taipei, Taiwan
g931303@hotmail.com

Yueh-Teng Hsu
LiteOn Technology Cooperation,
Taipei, Taiwan
Matthew.Hsu@liteon.com

***Abstract*—The total data rate of DVB (Digital Video Broadcasting) is 9.953Mbps much greater than DAB (Digital Audio Broadcasting). It is necessary to use better signal processing algorithm to improve the performance of DVB receiver. Hence, we propose some methods: (1) To optimize Reed-Solomon and Viterbi decoder; (2) To use parallel process instructions; (3) To Reduce FOR Loop and IF Branch, etc. After these modifications, the software radio in real-time reception based on PC can be implemented. In summary, we demonstrate a software receiver which operates at an acceptable real-time decoding speed.**

Keywords-software radio; mobility; Digital Video Broadcasting; orthogonal frequency-division multiplexing.

I. INTRODUCTION

The SR (Software Radio) based on PC platform has become popular and important in software GPS (Global Position system) [1] and software DAB (Digital Audio Broadcasting) [2]. Besides, the parallel process is one of the important and useful methods to enhance the decoding speed in SR. It is possible to use the software to implement the demodulation and decoding at real time based on PC. The total data rate of DAB is 2.4 Mbps (DQPSK); DVB-T (Digital Video Broadcasting-Terrestrial) is 9.953 Mbps (16QAM, guard interval 1/4, rate 2/3 convolutional code, non-hierarchical system for 6 MHz channels). Thus, the data rate of DVB-T is much greater than DAB. We hope to implement the SR which is used for DVB-T signal in Taiwan [3]. The advantages and benefits of SR are multimode and low cost. Besides, it is easy to maintain and modify. As we know, it is almost impossible to manufacture a new ASIC (Application-Specific Integrated Circuit) chip, which is relatively very expensive and time-consuming. On the contrary, the SR receiver does not need any additional hardware. The most important benefit of SR research, when compared with hardware radio, is that people can modify the signal processing procedure, algorithm and test the result easily. People can use the better detecting or estimation algorithm with software language to improve the performance of radio receiver rather than producing a new ASIC chip. For real-time DVB-T reception, we need to improve the slowest part first. Therefore, we have to implement the software of the RS (Reed-Solomon) and Viterbi decoder [4], which is fast enough to decode in real time. In order to enhance the SR decoding performance, we use the following ways: Looking up tables to reduce operation of program and SIMD (Single Instruction Multiple Data) instructions to decrease the CPU operation time [5], etc. Meanwhile, the performance of Viterbi decoder also needs to be improved by several methods: (1) To use SIMD instructions; (2) To reduce the branch of the program; (3) The other efficient methods [4].

This paper is organized as follows: We describe the previous SR in Section I. The proposed system architecture is described in details in Section II. Implementation and results are described in Section III. Section IV presents the conclusions.

II. SYSTEM ARCHITECTURE

The basic components for a software research platform include a PC and a radio front-end. The architecture is shown in Figure 1. It has hardware and software parts. The software structure is comprised of baseband processing, MPEG2 audio decoder and test application. The system memory acquires the digitized IF (Intermediate Frequency) signal from the tuner device by the driver; this is so called raw data. When enough raw data is collected, we can calculate the SNR and do baseband processing or dump that into the hard disk for future analysis. Thus, we can get the transport stream after baseband processing.

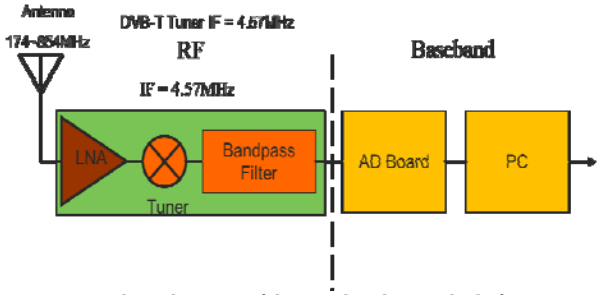


Figure 1. The architecture of the introduced research platform.

 ISBN: 978-1-61208-154-0

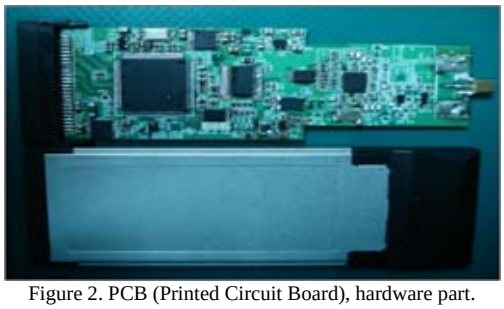


Figure 2. PCB (Printed Circuit Board), hardware part.

A. *Hardware*

The hardware part is shown in Figure 2. The receiver can decode the radio signal and record the baseband signal into hard disk simultaneously. It has several parts in our AD board. This device is comprised of A/D chip and LVDS (low voltage differential signaling) to reduce the transmission interference. The architecture is shown in Figure 3. The tuner module has relations with the receive sensitivity. Low IF signal output amplify gain is suitable for the requirement of analog to digital converter and so on. In our design, the silicon tuner (Xceive xc3028) is chosen to fit the requirements. The TLI 5540 is a high-frequency signal processing IC and converts the received frequency to IF frequency (4.571MHz). The frequency command is fed through the I^2C (I-squared-C) interface on this tuner. According to the DVB-T spec: the sampling rate of 8 MHz is 64 MHz /7= 9.142858 (MHz). Thus, we use the (64/7) x3 = 27.42857 (MHz) for 8bits resolution sampling rates to be digitalized by an AD converter and is fed into PCI-Express (bridge) chip (Transfer rate > 1 gigabyte per second) and can be sent to PC for future processing.

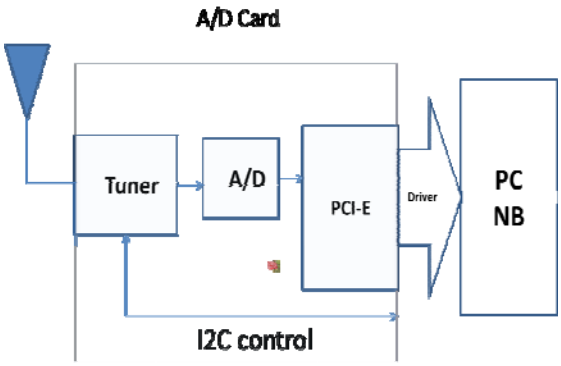


Figure 3. The architecture of the hardware part.

TABLE I. TUNER CONFIGURATION.

Parameter	Value
Frequency Input range	174~854 MHz
Bandwidth	6/7/8 MHz
Method of down conversion	Up down convertor

B. *Software*

The software has two parts. One is baseband receiver, and the other is baseband data recorder. The signal processing of software structure is shown in Figure 4 and 5, which is described as below: The purpose of time synchronization is used to synchronize the symbol in the FFT (Fast Fourier Transform) window correctly. We use the CP (Continual Pilot) to estimate integral frequency offset and use the TPS (Transmission Parameter Signaling) pilot to estimate the offset of OFDM (Orthogonal Frequency-Division Multiplexing) symbol. The Fractional frequency Synchronization can modify the signal after digital down converter in correct baseband frequency. We will use scatter pilot to estimate the channel coefficients. After doing fractional frequency synchronization and integer frequency synchronization, the residual frequency offset still exists. However, we can utilize a mathematic model for an offset with sampling frequency offset and residual frequency. We have to do channel estimation to compensate the phase error. The DVB-T system uses TPS to do frame synchronization.

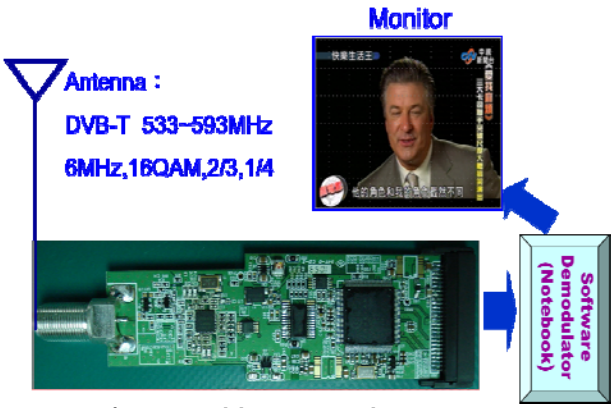


Figure 4. Software Demodulator operation diagram

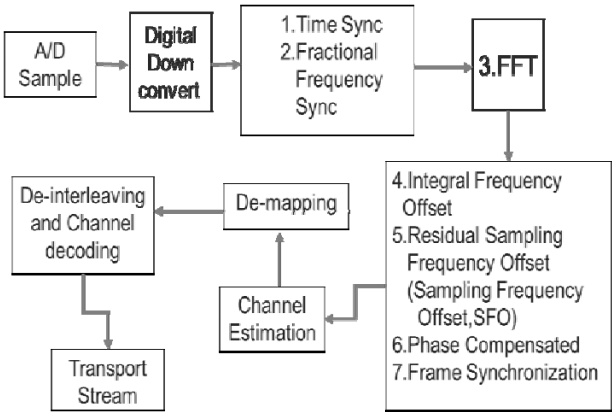


Figure 5. Structure of BDA(Broadcast Driver Architecture) filters

1) Sampling Frequency Offset: It will generate SFO (Sampling Frequency Offset) when the baseband sampling

rate of receiver cannot match the transmitter. Thus, we have to use some algorithms to modify SFO [6]-[7].

$$\delta = \frac{\delta_a}{T_s}$$
$$T_s' = T_s(1+\delta) \quad (1)$$

The u_{as} is total time shift of symbols, u_s is the quantization after periodic sampling time T_s'. The baseband symbol sampling signal is described as below:

$$r_s[n] = r(n \cdot T_s' + u_{as}) = r(n \cdot T_s \cdot (1+\delta) + u_{as})$$
$$= \sum_k X'[k] \cdot \exp\{ j2\pi \frac{k}{N} \cdot (1+\delta) \cdot n\}$$
$$X'[k] = X[k] \cdot \exp\{ j2\pi \frac{k}{N} u_s\} \quad (2)$$

The N means total numbers of carriers; N_g is the length of repeat signal. After demodulation (G_{imp} is the ICI caused by SFO. It can be ignored), we will have Rs (the r_s after FFT calculation):

$$R_s[k] = fft\{r_s[n]\} = X'[k] \cdot I_{k,k} + \sum_{g,g\neq k} X'[g] \cdot I_{g,k}$$
$$\cong X'[k] = X[k] \cdot \exp(j2\pi \frac{k}{N} u_s) \quad (3)$$

Furthermore, u_s is the total time shift with S_{th} symbols because of the periodic sampling frequency offset (caused by δ) . The function is given by:

$$u_s = u_{s-1} + \delta \cdot (N + N_g) \quad (4)$$

Assume the k as continual pilot. The $R_s[k]$ and $R_{s-1}[k]$ only has one phase error. The $\Delta\theta$ represents the phase error between the adjacent symbols. The SFO can be modified by using this relation between these two subcarriers. It can be expressed as follows:

$$\Delta\theta = angle(R_s[k]) - angle(R_{s-1}[k])$$
$$= 2\pi \frac{k}{N}(u_s - u_s - 1) = 2\pi \frac{k}{N} \delta \cdot (N + N_g) \quad (5)$$

After these calculations, it still has the residual subcarriers frequency offset. We have to consider the average phase shift. Finally, we can get the estimation of SFO as below:

$$\hat{\delta} \cong \underset{k}{mean}\{\frac{\Delta\theta_k - \varphi}{2\pi \cdot \frac{k}{N} \delta \cdot (N + N_g)}\}$$

Continual pilot

$$\varphi \cong \underset{k}{mean}\{\Delta\theta_k\} \quad (6)$$

2) Reed-Solomon decoder optimization: As we know, the RS decoder is the most time-consuming part of software DVB-T receiver. Hence, we want to improve the performance of RS decoder [5] which has four parts, and each part uses different algorithms that are described as below: First, there are 4064 additions and 4080 multiplications in GF (256) for getting syndrome:

$$S_0 = r(\alpha^0) = r_0 + r_1\alpha^0 + r_2\alpha^0 + \cdots + r_{254}\alpha^0$$
$$S_1 = r(\alpha^1) = r_0 + r_1\alpha^1 + r_2\alpha^2 + \cdots + r_{254}\alpha^{254}$$
$$S_2 = r(\alpha^2) = r_0 + r_1\alpha^2 + r_2\alpha^4 + \cdots + r_{254}\alpha^{253}$$
$$\vdots$$
$$S_{15} = r(\alpha^{15}) = r_0 + r_1\alpha^{15} + r_2\alpha^{30} + \cdots + r_{254}\alpha^{240}$$

1st vector table | 2nd vector table | 255th vector table

The lookup vector tables will be used to replace these operations. Thus, we create 255 vector tables; each table represents the all possible answers of 16 multiplications in GF (256). You can get 16 multiplications in GF (256) to look up vector table once. The multiplication of GF (256) is unnecessary. Second, we use the GF (256) with lookup vector tables and rewrite the whole Chien search program [8] within the assembly language to replace the multiplications. The performance of the Chien search program will be improved greatly by means of these modifications. It means that the decoding speed is much faster than the original Chien Search program.

TABLE Ⅱ. 2ND VECTOR TABLE

r_1	$r_1\alpha^0$	...	$r_1\alpha^{15}$
1	1	…	38
255	255	…	174

Third, to Reduce FOR Loop and IF Branch: The branch commands like IF, FOR, WHILE, etc., the branch prediction errors will cause the speed of execution down. We need to reduce all the loops in our program by means of using a large number of C or assembly code. Thus, we rewrite some program to generate those C or assembly codes. We write a program to produce a large C code to expand it.

 ISBN: 978-1-61208-154-0

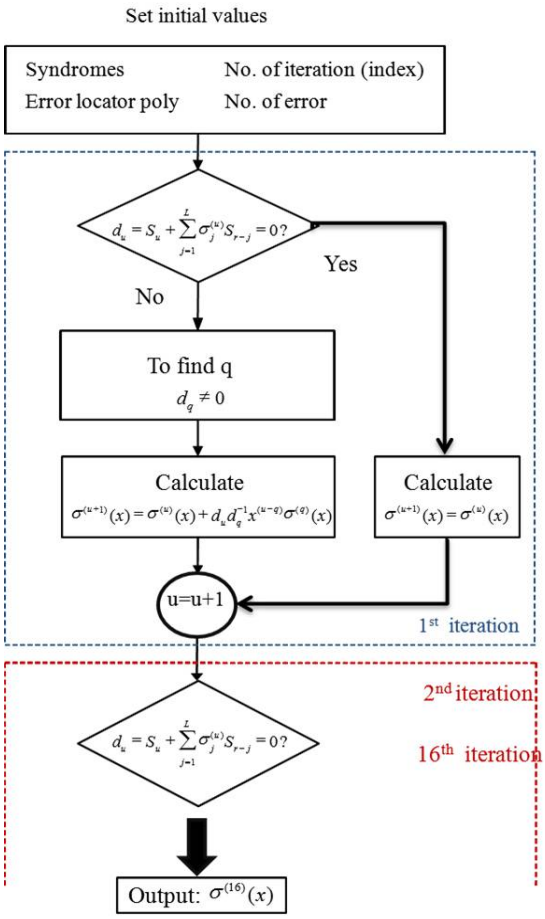


Figure 6. The modified procedure of BM(Berlekamp-Massey) [6] in our program

Besides, the 64-bit CPU provides sixteen XMM registers. It is enough and useful to place all operational values of Forney algorithm procedures by using these registers. It can reduce the slow operation of transferring data between registers and memories as much as possible. The data are only moved among registers.

3) *Viterbi decoder and sytematic approach of software optimization:* Reed-Solomon decoder optimization: In order to speed-up the Viterbi decoder program for software DVB-T receiver. We use the Explorer in Microsoft Visual Studio 2008 (Team Suite edition) to find out and analyze the slowest parts of our programs. Hence, we propose various ways to improve the performance which are described as below [4]: The Viterbi decoder consists of several units: BMU (Branch Metric Unit), ASCU (Add-Compare-Select Unit), PMMU (Path Metric Memory Unit), and SMU (Survivor Memory Unit). The results of the channel estimation also are used to help the decoder to improve the BER (Bit Error Rate). Meanwhile, in order to improve our decoder performance we need some useful optimization methods. The procedures of the optimization are shown in Figure 7.

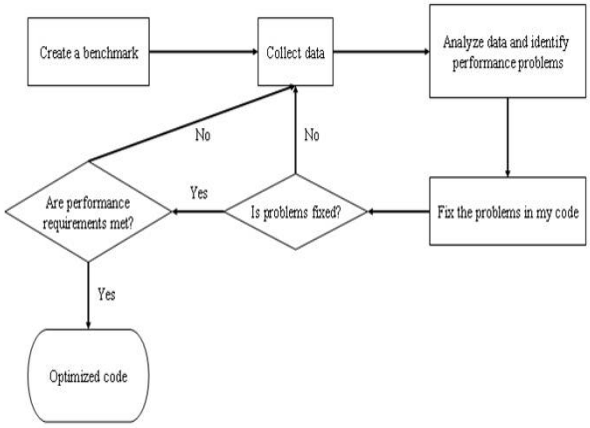


Figure 7. The systematic approach of software optimization.

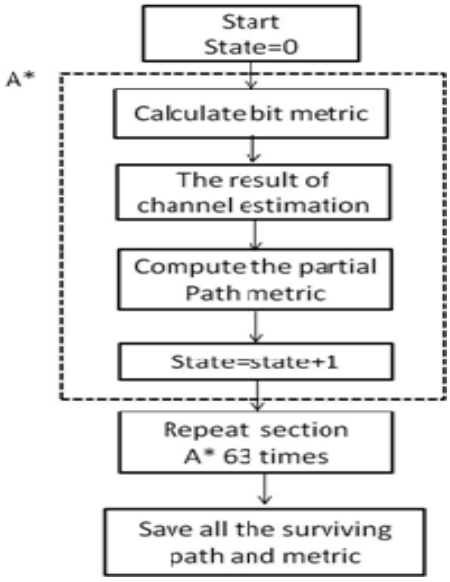


Figure 8. The flowchart of the modified program. A loop is expanded to 64 identical segments A*.

C. SSE4 Instuctions

The newest Intel CPU has the SSE4 (Streaming SIMD Extension 4) instructions [4]. We use the new Intel CPU SIMD instructions to be our tool in order to save the CPU operation time. We can combine it by making some operations of these instructions, and it will be much more efficiency than before.

The execution efficiency of the procedure will drop greatly when containing the judgment operations in the procedure (like IF, FOR, WHILE, etc.).

The efficiency of the procedures would be improved greatly by reducing these operations in the program. General speaking, the FOR loop in C program is used to deal with the continued and similar procedure. In our original Viterbi decoder, the program has many FOR loops. Hence, we create

 ISBN: 978-1-61208-154-0

a lot number of C or assembly code in order to reduce all the loops in our program. (The new procedure is shown in Figure 8).

D. The Other Efficient Methods

The same result can often be got through different ways when writing the program. Therefore, we have to find out or design the new algorithm to speed up the operation of procedures. In fact, there are some other structures that can be used to replace the conventional structures. For example, the C code can be improved by changing the structure. The every 2 coded bits contains only 4 types of bit metric and 2 types of channel estimation results. Therefore, the structure can be modified by combining these computations (shown in Figure 9). The assembly and SSE instructions can implement it.

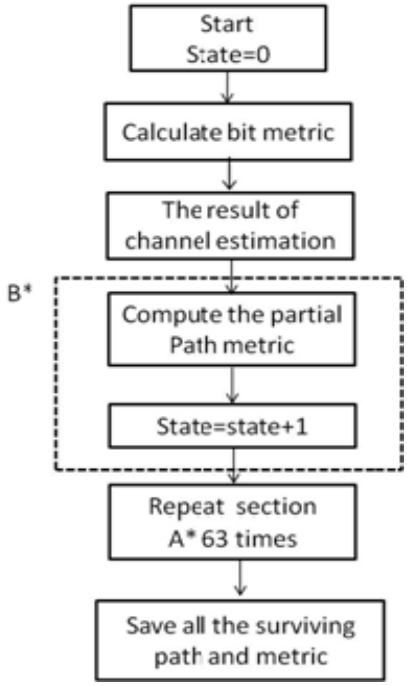


Figure 9. The flowchart of the simplified program. The bit metric calculation and channel estimate multiplication are not repeated 64 times as in Figure 8.

E. Rewrite The Whole Viterbi Coder Program With Assenbly Language

The 64-bit CPU provides more registers to reduce the operation of moving data. The x32 compiler is only allowed to use 8 of 16 XMM registers in the 64 bit CPU. Hence, we have to use all XMM registers by using x64 compiler. Furthermore, we can use the sixteen to reduce the operation time of moving data from registers to memories or moving data from memories to registers. But it is not allowed to use inline assembly by using x64 compiler. The whole programs need to be rewritten within assembly language to adapt to the x64 compiler. The additional eight XMM registers will be used to hold the metric data in order to reduce the movements between memories and registers.

III. IMPLEMENTATION AND RESULTS

The environment of purposed receiver is a desktop PC the specification of which is listed in Table Ⅲ. The OS is windows 7. The software languages are C, C++, C#, or Assembly.

The programming tools are mainly from Microsoft and Intel including compiler and mathematical library.

TABLE Ⅲ. PC PLATFORM SPECIFICATION INCLUDING OS.

Component	Spec.
OS	Windows 7 (64bit)
CPU	Intel(R) Core(TM) i7-2600K CPU @ 3.40GHz (8 CPUs), ~3.4GHz
RAM	4.0 GB
Motherboard	ASUS P8H67-M PRO(REV3.0)

In this research, the name, usage and vender for all tools are listed in Table Ⅳ. The CPU loading of each individual block is listed in Table Ⅴ. Furthermore, The DVB-T signal is also used to input our proposed system. We can get the final results shown in Figure 10. According to the results, it has no interference in this figure.

TABLE Ⅳ. DEVELOPMENT TOOLS.

Tool	Vender	Using for
Visual studio. Net	Microsoft	Compiler, Application
Windows Platform SDK	Microsoft	Driver
DirectShow	Microsoft	API

TABLE Ⅴ. THE CPU LOADING OF EACH BLOCK

Block	Elapsed Inclusive Time (ms)
Time & Frequency Synchronize	63.66
Remove CP & FFT	67.82
Channel estimation	145.87
Deinner & Depuncher	54.49
Deoutter interleave	17.93
Demodulator	8.93
Viterbi decoder	1096.79
RS Decoder	30.67
Descrambler	0.92
Frame Synchronize	8.58
Program Initialization	27.52
Phase Compensation	8.44
C++ standard library	10.27
total	1541

 ISBN: 978-1-61208-154-0

Figure 10. Software Demodulator Result

IV. CONCLUSION

As we know, the real-time bit rate of DVB-T is 9.953 Mbps (16 QAM, guard interval 1/4, rate 2/3 convolutional code, non-hierarchical system for 6 MHz channels) in Taiwan. In this research, we utilize lookup tables, SIMD instructions, loop expansion and the other efficient methods to greatly improve the decoding speed of our Reed-Solomon and Viterbi decoder. The decoding rate of the proposed system is more than the required bit rate of the real-time DVB-T. So, our system is fast enough to process the DVB-T signal used in Taiwan in real-time.

The proposed system is easy to operate with the current commercial PC and it helps us to develop new baseband algorithm. Therefore, we can verify it in real-world environment. It only takes 1541ms to decode the 2760ms video data in our research. Thus, the real-time software DVB-T receiver is possible to be accomplished after these proposed modifications.

REFERENCES

[1] N. Kubo, S. Kondo, and A. Yasuda, “Evaluation of code multipath mitigation using a software GPS receiver,” IEICE Trans. Commun., vol. E88-B, no. 11, pp. 4204-4211, Nov. 2005.

[2] Shu-Ming Tseng, Yueh-Teng Hsu, Meng-Chou Chang, and Hsiao-Lung CHAN, “A Notebook PC Based Real-Time Software Radio DAB Receiver, ” IEICE Trans. Commun., vol. E89-B, no. 12, pp. 3208-3214, Dec. 2006.

[3] ETSI EN 300 744 :Digital Video Broadcasting (DVB); Framing structure, channel coding and modulation for digital terrestrial television ETSI, 2004.

[4] Shu-Ming Tseng,Yu-Chin Kuo,Yen-Chih Ku, and Yueh-Teng Hsu, “Software Viterbi Decoder with SSE4 Parallel Processing Instructions for Software DVB-T Receiver, ” in Proc. The 7th IEEE International Symposium on Parallel and Distributed Processing with Applications (ISPA-09), Aug. 2009, pp. 102-105.

[5] Shu-Ming Tseng, Jian-Cheng Yu, Jheng-Zong Shih, and Yueh-Teng Hsu, “Reed-Solomon Decoder Optimization for PC-Based DVB-T Software Radio Receiver, ” in Proc. The International Conference on Consumer electronics, Jan. 2011, pp. 393-394.

[6] Michael Speth, Stefan Fechtel, Gunnar Fock, and Heinrich Meyr, “Optimum Receiver Design for OFDM-Based Broadband Transmission—Part II: A Case Study, ” IEEE Trans. Commun., vol. 49, no. 4, pp. 571-578, Apr. 2001.

[7] Michael Speth, Stefan Fechtel, Gunnar Fock, and Heinrich Meyr, “Optimum Receiver Design for Wireless Broad-Band Systems Using OFDM—Part I: A Case Study, ” IEEE Trans. Commun., vol. 47, no. 11, pp. 1668-1677, Nov. 1999.

[8] S. B. Wicker, Error Control Systems for Digital Communications and Strorage, Prentice Hall, 1995.

 ISBN: 978-1-61208-154-0

Evaluation of Seed Throwing Optimization Meta Heuristic in Terms of Performance and Parallelizability

Oliver Weede Stefan Zimmermann Björn Hein Heinz Wörn
Institute for Process Control and Robotics (IPR)
Karlsruhe Institute of Technology (KIT)
Karlsruhe, Germany
e-mail: {oliver.weede, stefan.zimmermann2, bjoern.hein, woern}@kit.edu

***Abstract*— Seed Throwing Optimization is an easy to implement probabilistic metaheuristic for multimodal function optimization with roots in hill climbing and the evolutionary computation like technique Harmony Search. It is a randomized Gradient Ascent with multiple initial states and the possibility to limit exploration to only paths which have shown potential. In this paper, the speed of convergence of Seed Throwing Optimization is compared to Multi-Level Gradient Ascent, Harmony Search, Particle Swarm Optimization and Simulated Annealing. Improvements of the Seed Throwing Optimization are presented. Parallelizability of the mentioned metaheuristics is examined. Parts which are suitable for parallelization are extracted by identifying data and control flow dependencies. Two applications, port optimization in minimally invasive surgery and network parameter optimization for a distributed robotic system, are shown. The presented methaheuristics are tested in a benchmark. The highest convergence speed could be achieved with Harmony Search and Seed Throwing Optimization.**

Keywords-Multimodal function optimization; Parallel computing; Port optimization in minimally invasive surgery; Network optimization.

I. INTRODUCTION

Probabilistic metaheuristics are capable of solving very generalized classes of problems in many future computational applications. A metaheuristic is able to find a near optimum solution of a multi modal optimization problem in an efficient way. If a good solution of a multi modal function is found, it is not known if the solution is globally good. Thus the main problem that a metaheuristic has to deal with is to avoid premature convergence to a local optimum. If there is more than one good solution the issues arise of how to locate all (or some) of these solutions. There is a trade-off between *diversification* and *intensification*. Diversification refers to the exploration of the whole search space, intensification to the exploitation of the accumulated experience. The speed of convergence depends on the right balance of these forces [1].

Many approaches have been developed combining or enhancing existing methods, an overview and a taxonomy of global optimization methods can be found e.g. in Weise et al. [2], but there is no optimal solver for any optimization problem. Thus the best way is to use problem specific knowledge to choose an appropriate algorithm, convenient constraints to limit the search space or to modify strategies of an algorithm. Thus a further criterion of an optimization algorithm is its simplicity, which means that it is easy to understand and to implement.

A focus in this paper is Seed Throwing Optimization (STO), developed in 2009 [3]. It is a randomized Gradient Ascent with multi initial states and the possibility to explore only paths which have shown to be good. STO is a probabilistic metaheuristic based on the paradigm of an iterative local search. Thus, in each step a first solution is selected, a neighborhood of "similar" solutions is defined and the best solution of the neighborhood is determined. The idea of STO is to choose iteratively initial seeds over the domain of the objective function and to spread out more seeds in the neighborhood respecting the direction of the steepest ascent. Initial seeds are taken randomly or form a current best memory and are used for further exploration. The number of seeds "thrown out" from the initial point is controlled by the value of the function. Large values leads to extensive exploration. The neighborhood that is explored is like a gradient ray, but larger spreads have the form of a fan.

Parallelizability is another important criterion since clusters and multi-core processor architectures are commonly available and promise high acceleration. By splitting up algorithms appropriate high performance can be achieved. We compare parallelizability of several metaheuristics. In many real-life scenarios, objective functions are expensive to compute. In order to examine and classify STO in its potential parallelization degree, we have to identify parts of the algorithm that are suitable for parallelization. The pivot of successful parallelization is to avoid data and control dependencies between consecutive instructions that occur in the execution of the algorithm. For this, the parallelization of the algorithms will be discussed upon the pseudo code algorithm formulations introduced in this paper.

The objective function is treated as a black-box, no deeper insight of the characteristics is needed (direct optimization).

II. APPLICATIONS

We use STO in two applications that are shortly described in this section.

A. Port Optimization in Minimally Invasive Surgery

In minimally invasive surgery surgical instruments and an endoscopic camera are injected through so called trocars. The placement of the trocars is an important factor for the success of an intervention. The da Vinci® Surgical System

 ISBN: 978-1-61208-154-0

[4] is a telemanipulation system which is clinically used to perform interventions more ergonomic and precise. The manipulator is controlling the surgical instruments and the endoscope. Beside the trocar placement the selection of an appropriate initial pose of the manipulator is important. It contains the configuration of the stationary joints and the position and orientation of the manipulators base**.** Poor choices of the manipulators pose or the placement of the trocars can lead to collisions of the arms or unreachable target areas [12].

Finding optimal trocar positions and an optimal manipulator´s pose can be formulated as optimization problem consisting of active- and passive constraints and a goodness measure to be maximized. The passive constraint is corresponding to the trocar positions (ports), which remain stationary throughout the intervention. The pivot point of the surgical instruments has to be close to the chosen port positions. The active constraints refer to the trajectory of the end-effectors of the surgical instruments. Each target has to be reached. A configuration is rejected, if the passive constraints are not fulfilled (objective function is set to zero). If a target is unreachable or collisions occur the current configuration is rejected too. The objective function is a conjunction of several components. Separation of each arm to other arms, preoperatively segmented zones of risk and the bones are used as the main component of the objective function. Tilting of the table and ergonomic reasons like manipulation angle between the working instruments also influence this function. An evaluation of the objective function is very time consuming because minimal distances in a virtual scene containing over 200 000 triangles have to be computed for each point of the trajectory.

The result of the optimization is transferred to the operation theatre by using methods of augmented reality. The port positions are projected directly on the patient´s abdomen. The optimized pose of the manipulator is shown by a virtual scene containing a model of the manipulator and arrows to indicate the optimized pose.

B. Network parameter optimization for a distributed robotic system

State-of-the-art autonomous intuitive assistance and information systems for service robots compensate for the tremendous loss of information for remote users. Often augmented and/or virtual reality is used to provide immersion to the remote environment. A guaranteed minimum performance of the communication between the robotic system and the remote user is needed in order to still guarantee a suitable degree of immersion. Using non-dedicated standard means of communication, e.g. the Internet, makes constraints as audio and video synchronicity (<200ms) or latency for tactile tasks (<180ms) hard to fulfill [9]. A vast amount of multimedial and multimodal data has to be transmitted to provide a high degree of immersion. Thus, an optimization of the data flows in the robotic system as well as between the robotic system and the remote user is required to provide maximum immersion.

Based on the fact that there's no access to management functions of any transmitting node in the network for this scenario, two common approaches have been examined: (i) optimized individual data flows on the sending endpoints and (ii) traffic shapers on the sending endpoints.

In the first case, for every individual data stream measures as packet size and latency as well as global bandwidth and net load have to be optimized for a given network topology of a distributed robotic system and different kinds and severities of perturbations. In the second case, parameters for the traffic shapers like Hierarchical Token Bucket (priority, bandwidth, etc.) or Completely Fair Queuing have to be found. This is achieved by a near-reality test that runs the traffic of slightly modified original software components through the network simulator NS3 [11] that provides the internal network topology of the robotic system and the Internet as well as the perturbations.

The recorded sequence numbers and time stamps of the packages gained from the simulator run is then evaluated by an objective function script. The objective function is weighting the rate of arriving control commands (e.g. for robots) more than the rest of data streams (e.g. audio or video). The data stream throughput/quality and the inverse of the latency of packages also contributes to the objective function. If packages are lost the value of the objective function is decreased. These are the most important parameters which determine the configuration of the streams and/or the traffic shapers.

III. COMPARED METAHEURISTICS

In this section the five metaheuristics are shortly described.

C. Multi-level Gradient Ascent

Gradient Descent/Ascent figuratively speaking follows the graph from state to state, always locally increasing the value as much as possible by the updating rule

$$\mathbf{x}_{k+1} = \mathbf{x}_k + h\, \nabla \mathrm{f}(\boldsymbol{x}_k). \qquad (1)$$

The function could be treated as a black-box and finite differences could be used to approximate the steepest ascent. If the parameter $\boldsymbol{x}$ of the objective function has two dimensions, the function can locally be approximated by a plane. In this plane the steepest ascent can be computed.

Therefore, two points $\boldsymbol{b}$ and $\boldsymbol{c}$ in the local neighborhood of an initial point $\mathbf{a}=(a_1, a_2, a_3)^{\mathrm{T}}$ are chosen. Point $\mathbf{a}$ is the current best solution of the k-th iteration ($\mathbf{x}_k$ in eq. (1)). The three points determine the plane. With a parameter α in the real interval [0, 2π] a circle on the plane is defined by the function

$$\mathbf{r}(\alpha) = (\mathbf{b} - \mathbf{a}) \sin \alpha + (\mathbf{c} - \mathbf{a}) \cos \alpha. \qquad (2)$$

To determine the steepest ascent, the first and the second deviation of the third component of $\boldsymbol{r}(\alpha)$ is regarded. By setting

$$\partial_\alpha\, r_3(\alpha) = 0 \text{ and } \partial^2_\alpha\, r_3(\alpha) < 0 \qquad (3)$$

```
(1)  Define f(x), x = (x1, x2, …, xN)^T
(2)  step width h_rought
(3)  infinitesimal step width h_inf
(4)  for all levels
(5)    h_rough = max( h_rough/2, h_inf)
(6)    select random solution x
(7)    p_start = GA(x, h_rough)
(8)    q = GA(p_start, h_inf)
(9)    remember q, if better
(10) end for
(11)
(12) function GA(starting point x_k, step width h)
(13) do
(14)   x_k+1 = x_k + h∇f(x_k)
(15)   evaluate function f(x_k+1)
(16) while (f(x_k+1) > f(x_k) or max. number of iterations reached)
(17) return x_k+1
```

Figure 1. Pseudo code of Multi-level Gradient Ascent (MLG)

the direction α of the steepest ascent in the plane can be computed. The solution for α is published in [3] (Eq. 4-7). With this method it is possible to approximate the gradient by only evaluating *one* further function value in each step. The points **b** and **c** out of step k can be used to define the plane for $\mathbf{x}_{k+1}$.

Gradient ascent only reaches an arbitrary optimum, not necessarily the global one. In *Multi-Level Gradient Ascent* several gradient ascents are performed with various initial states. Decimating the function in a way that only function values of a rough grid are chosen leads to a low pass filtering and many local optima are filtered out. In each level a gradient ascent is performed on the undersampled function and the result is used for a gradient ascent with a subtle grid. The mesh size of the rough grid is iteratively decreased. Fig. 1 shows the pseudo code of Multi-level Gradient Ascent.

D. Harmony Search

```
(1)  Define f(x), x = (x1, x2, …, xN)^T
(2)  accepting rate r_accept
(3)  pitch adjusting rate r_p
(4)  pitch variation range p_range
(5)  Generate best memory with B_size random solutions
(6)  while (maximum numbers of iterations not reached)
(7)    for all N components x_i
(8)      if (rand() < r_accept) choose value from best memory for x_i
(9)        if (rand() < r_p) x_i = x_i + r p_range with rand. val. r in [-1,1]
(10)     else
(11)       choose a random value for x_i
(12)     end if
(13)   end for
(14)   evaluate function f(x)
(15)   remember solution x in best memory, if better
(16) end while
```

Figure 2. Pseudo code of Harmony Search (HS)

Geem et al. [5] developed an optimization algorithm which is inspired by the improvisation process of a musician. Fig. 2 shows the pseudo code. rand() always denotes a function that returns equally distributed random numbers in the [0,1]-interval.

```
(1)  Define f(x), x = (x1, x2, …, xN)^T
(2)  depth of local search r_loc
(3)  maximum number of seeds thrown out n_MaxSeeds
(4)  Initialize best memory B with B_size random solutions
(5)  while (maximum numbers of iterations not reached)
(6)    if (rand() < 0.5) choose initial seed x_0 randomly
(7)    else choose seed x_0 from the best memory
(8)    endif
(9)    evaluate function f(x_0)
(10)   compute direction α of steepest ascent in x_0
(11)   compute number of seeds n_x (Eq. 5)
(12)   for each seed
(13)     compute position of seed x_i (Eq. 6)
(14)     evaluate function f(x_i)
(15)   end for
(16)   remember solution in best memory if better
(17) end while
```

Figure 3. Pseudo code of Seed Throwing Optimization (STO)

E. Seed Throwing Optimization

Seed Throwing Optimization (Fig. 3) has its roots in Gradient Ascent and Harmony Search.

For initializing the algorithm, the objective function f(**x**), $\mathbf{x} = (x_1, x_2, \ldots, x_N)^T$ and its domain is defined. The parameter B_{size} determines how many current best solutions are remembered in a (B_{size}, N)-matrix **B**. The size of the matrix should be chosen relatively small, thus a high intensification is ensured. The matrix is initialized with random values or, if known, with initial guesses for the solutions. The objective function is evaluated at these points. Associated to this matrix there is a B_{size}-vector containing the function values of the solutions. At every function evaluation during the runtime, the maximum and minimum values of the objective function are remembered within the variables g_{min} and g_{max}. The parameter r_{loc} controls the local search radius for initial seeds. We suggest taking the maximum interval of the domain divided by four. Choosing a big value leads to diversification, small values to intensification.

The main loop starts by selecting an initial seed $\mathbf{x}_0$. If a random number is below a threshold, an initial seed is chosen randomly; otherwise it is taken out of the matrix ***B*** which stores the best solutions. Like in Harmony Search this procedure could be done for each component of the solution to enable mutations of a known good solution. After evaluating f($\boldsymbol{x}_0$) an intensification factor n_x is computed. It determines how many seeds will be used to explore the neighborhood of $\boldsymbol{x}_0$. The higher the value f($\boldsymbol{x}_0$) the more seeds are used. Thus good solutions are explored more than valleys. We suggest to use

$$n_x = n_{MaxSeeds} \left\lfloor \frac{f(\mathbf{x}_0) - g_{Min}}{g_{Max} - g_{Min}} \right\rfloor . \qquad (5)$$

with the parameter $n_{MaxSeeds}$=5, which determines the maximal number of seeds "thrown out". The parameter n_x depends on the current maximum g_{max} and the current minimum g_{min} because the range of values of the objective function is not known. The parameter n_x is an intensification factor that controls the number of repetitions of the inner

loop (lines 12-15) of the algorithm. Two components of $\mathbf{x}_0$ are chosen randomly (e.g. x_1 and x_2). The direction of the steepest ascent for these chosen dimensions is computed (Eq. 4-7 in [3]). Let it be α_0, then $\mathbf{r}(\alpha_0)$ (Eq. 2) is the direction of the steepest ascent. Let $\mathbf{d}_\alpha$ be the normalized vector in this direction, and let $\mathbf{d}_\varphi$ be a normalized vector in a random direction φ. Than the position of seed i, which is "thrown out" from the initial seed, is determined by a convex combination of these vectors using two random numbers r_1 in [0.5,1] and r_2 in [0,1].

$$\mathbf{x}_i = \mathbf{x}_0 + r_2\, r_{loc}\, (\, r_1\, \mathbf{d}_\alpha + (1 - r_1)\, \mathbf{d}_\varphi\,). \qquad (6)$$

The function value of each seed $f(\boldsymbol{x}_i)$ is evaluated. If this solution is better than the worst solution in **B**, the new solution $\boldsymbol{x}_i$ replaces the old one and the new seed with its neighborhood has the potential to be further explored in a following iteration. If the maximum number of iterations is reached, the best solution of **B** is taken as the global best solution.

The performance can be slightly improved, if each time a new solution is entering the best memory a mutation of solutions is done. We suggest two ways of *mutation*:

Select the nearest solution $\mathbf{x}_b$ of the best memory to the new solution $\boldsymbol{x}_i$ in terms of Euclidean distance and evaluate the objective function at the convex combination of both solutions: Accept the resulting solution

$$\mathbf{x}_m = \frac{f(\mathbf{x}_b)\mathbf{x}_b + f(\mathbf{x}_i)\mathbf{x}_i}{f(\mathbf{x}_b) + f(\mathbf{x}_i)} \qquad (7)$$

if $f(\mathbf{x}_m)$ is better than the worst one in the best memory.

The other way of mutation is to evaluate the convex combination of all solutions of the best memory. Let $\mathbf{b}_1,\ldots,\mathbf{b}_{Bsize}$ be the solutions (rows) of matrix **B**. Then

$$\mathbf{x}_m = \frac{\sum_{i=1}^{B_{size}} f(\mathbf{x}_i)\mathbf{x}_i}{\sum_{i=1}^{B_{size}} f(\mathbf{x}_i)} \qquad (8)$$

replaces the worst solution, if its function value is better.

A possibility to further improve STO is to measure the similarity of a new solution to the most similar solution in the best memory in terms of Euclidean distance. If the distance is below a threshold, then the new solution should replace the most similar solution to ensure diversity of the best solutions.

In high-dimensional optimization problems the last dimensions which improved the global solution can be remembered and can then be preferred for further exploration, instead of randomly choosing two dimensions.

F. Particle Swarm Optimization

A description of Particle Swarm Optimization can be found in [6]. Fig. 4 shows the pseudo code.

```
(1)  Define f(x), x = (x1, x2, ..., xN)^T
(2)  Generate particles p_i with random solutions and velocity v_i=0
(3)  Initialize best solution for each particle p_best,i := p_i
(4)  Initialize best solution of all particles g_best
(5)  while (maximum numbers of iterations not reached)
(6)    for all particles i
(7)      compute vector to best solution of particle p_dir,i = p_best,i - p_i
(8)      compute vector to global best solution g_dir,i = g_best - p_i
(9)      determine random values r1, r2 in [0,1]
(10)     update velocity v_i = v_i + c1 r1 p_dir,i + c2 r2 g_dir,i
(11)     update solution p_i = p_i + v_i
(12)     evaluate function f(p_i)
(13)     remember best solution of particle p_best,i
(14)     remember global best solution g_best
(15)   end for
(16) end while
```

Figure 4. Pseudo code of Particle Swarm Optimization (PSO)

G. Simulated Annealing

Simulated Annealing is introduced in [7]. The pseudo-code is shown in Fig. 5.

```
(1)  Define f(x), x = (x1, x2, ..., xN)^T.
(2)  initial temperature T0
(3)  choose initial solution p and evaluate f(p)
(4)  step width h for neighborhood
(5)  while (t < maximum numbers of iterations)
(6)    choose point q in neighborhood of p
(7)    evaluate function f(q)
(8)    calc. current temperature temp, e.g. T0 c^t, 0<c<1 or T0 log(t+2)
(9)    while (max. num. of iters not reached AND q not accepted)
(10)     if (exp( f(q) – f(p)) / temp <= rand() or f(p) < f(q))
(11)       p=q (accept new solution q)
(12)     end if
(13)   end while
(14)   remember solution q if better than current best solution
(15) end while
```

Figure 5. Pseudo code of Simulated Annealing (SA)

IV. PARALLEL COMPUTATION TECHNIQUES

Although hardware models can be used to a great benefit for the practical application of parallelization, for our purpose, discussing the parallelization on a level of pseudo code algorithm formulations is sufficient. This way, the pivotal problem of identifying parts of the algorithms which can be computed parallel has to be solved and this favors brevity a lot.

As we assume no means to estimate the time required for an evaluation of the objective function, it has to halt for any parameter in the range examined in the optimization. The computing time spent executing the code for the optimization algorithms themselves is negligible in comparison to the computing time spent for the evaluation of the objective function.

Some work has been done in this area mostly for the popular Simulated Annealing and Particle Swarm

 ISBN: 978-1-61208-154-0

Optimization (PSO) algorithms. A taxonomy of parallelization techniques and their usage for Simulated Annealing as well as the effect on convergence caused by asynchronously parallelizing Simulated Annealing are discussed and presented in [8]. A simple approach to profit from wide-spread low-cost computing clusters using a Message Passing Interface (MPI) is shown in [8].

As the objective function is much more expensive to compute than the optimization algorithms themselves, only the ability to parallelize the objective function invocations is examined. The main goal is to fill the task pool sufficiently such that (i) the maximum of data-dependent tasks is available at the time and (ii) the data-independent tasks end about the same time, the data-dependency ends.

There are synchronous and asynchronous techniques for parallelization. Synchronous parallelization maintains the order of execution of a serial algorithm. Thus, synchronization points are required to ensure that all needed data is available. So workers (process internal or external threads) might wait for other workers to finish. It maintains the convergence behavior of the non-parallelized algorithms in terms of objective function calls. In contrast to synchronous parallelization, asynchronous parallelization does not force synchronization points that are blocking workers. It is favoring speed over using the most recent data from other workers and determinism to overcome Amdahls Law.

First, just the possibility for synchronous parallelization is described.

H. Synchronous Parallelization

Most time consuming parts of *Multi-level Gradient Ascent* are the function evaluations in the GD function (lines 12-17). There, two objective function evaluations (line 14 and 15) are needed in each iteration. These computations cannot be run in parallel since line 15 needs the result of line 14 and in the next iteration the last two solutions have to be known. Nevertheless, multiple levels running the GD function can be executed in parallel (line 4-10). The number of function evaluations in each level differs significantly, thus early finishing workers idle before completion of a level.

In every iteration of *Harmony Search*, a single new solution is generated (inner loop, lines 7-13) which is to be explored. Since this newly gained function value is compared to the values of the best memory each succeeding iteration depends upon the iteration before and therefore the iterations cannot be parallelized synchronously. Just the B_{size} solutions for initializing the best memory can be executed in parallel (line 5).

For *Simulated Annealing* just the first function evaluation (line 3) and the first evaluation of a new solution q (line 7) can be parallelized. Each further iteration (lines 5-13) depends on an initial state that is stored in the last iteration (line 11).

In *Particle Swarm Optimization* just the initialization of the particles (line 2) can be executed in parallel. If the number of particles is a multiple of the number of workers, the idle time of the workers is minimal. The particle loop (lines 6-15) cannot be executed in parallel by strict synchronous parallelization, because each particle can alter the current global best solution (line 12): to any of the lines considering the global maximum).

Seed Throwing Optimization has two major time-consuming code parts: (i) the ascent determining part in line 10 (computation of the steepest ascent) and (ii) the evaluation of the "thrown out" seeds before remembering best solutions and moving on to the next iteration (lines 12-15). In the ascent determining part three function evaluations can be run in parallel, and the function evaluations in the seed evaluation part can be run in parallel. It is ideal when the number of evaluated seeds is a multiple of the number of workers. Proceeding after each of these two parts requires all parallel function evaluations of the iteration to finish. Thus, calculating the function evaluation in parallel takes as long as the function evaluation with the longest computing time.

I. Asynchronous Parallelization

For increased speed-up, the parallel metaheuristics can be made asynchronous, by weakening the data dependencies by definition to avoid any synchronization delays. The workers then exchange needed data, which can be out-dated at the time they are used. The modified asynchronous algorithm could not only benefit from less synchronization delays but can even benefit from using its old data, as stated for Simulated Annealing in [7]. If this is the case, it indicates that there is no balance of intensification and diversification and the currently explored area is just a local maximum, not the global one. Nevertheless first experimental results have shown that using slightly obsolete data has little to no effect on the convergence of the algorithms.

In *Simulated Annealing* a point in the neighborhood is selected (line 6) and is then possibly accepted within a random walk (lines 9-13). One approach to parallelize the algorithm asynchronously is to run these steps in parallel by different workers. Since these steps are not deterministic new points are examined. In contrast, *Multi-Level Gradient Ascent* is deterministic and therefore does not examine new points. But Multi-Level Ascent can be parallelized asynchronously by starting work on a new level, each time a worker has finished a level.

PSO can easily be converted to an asynchronous version. Each time a particle has reached a new global optimum or its local optimum, the information is stored in shared memory. Thus, the particles can evaluate the function in parallel and information about the global maximum is, if so, just slightly obsolete. The same approach can be used for asynchronous *Harmony Search*. The best memory is stored in the shared memory, every time a better solution is found.

STO is very well suited for asynchronous parallelization due to the fact that potentially half the iterations use a random seeding point (line 6-8). These iterations are independent among themselves. The only dependency exists to the other half of the iterations using best memory-dependent seeding points. Thus, while providing an update attempt for the best memory for about every second iteration of the data-dependent case, the convergence of asynchronous STO is very much the same as serial STO. Each time a worker idles a random seeding point case can be performed.

 ISBN: 978-1-61208-154-0

V. EVALUATION

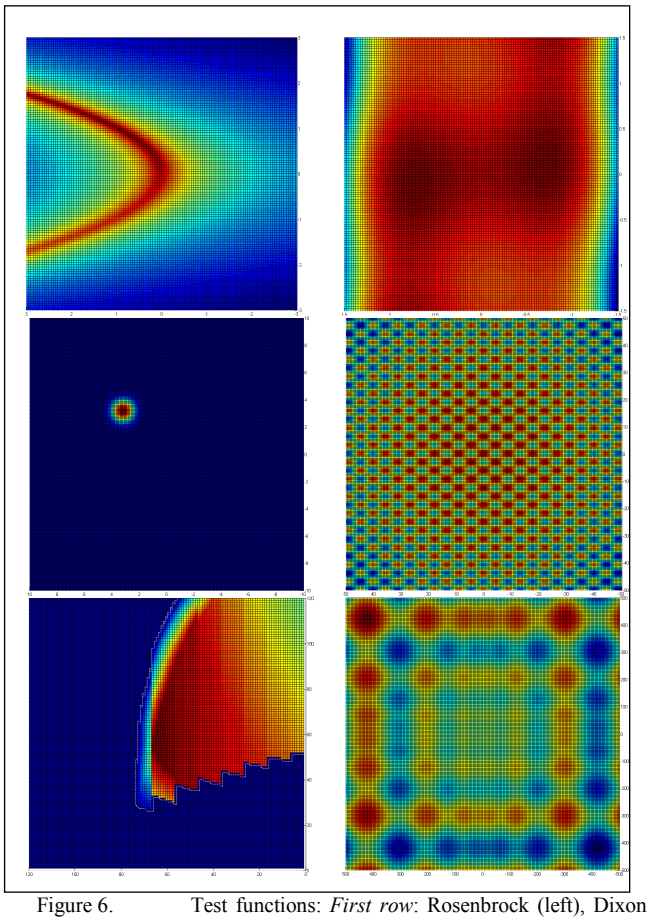


Figure 6. Test functions: *First row*: Rosenbrock (left), Dixon (right); *Second row*: Easom, Griewangk; *Third row:* Port, Schwefel. Maxima are colored in red.

An overview over the test functions, Rosenbrock´s Banana function, Dixon´s Camelback, Easom´s function, Griewangk´s and Schwefel´s function, can be found in [10]. We also tested the port optimization problem in minimally invasive surgery for the right-hand instrument. Fig. 6 shows the test functions and Table I the formulas and domains of the functions.

TABLE I. TEST FUNCTIONS, NAME, DOMAIN

$-\log\left(1+(1-x_1)^2+100\left(x_2-x_1^{\,2}\right)^2\right)+10$	Rosenbrock $[-3, 3]^2$
$-\left(4-2.1x_1^{\,2}+\frac{x_1^{\,4}}{3}\right)x_1^{\,2}-x_1x_2-4\left(x_2^{\,2}-1\right)x_2^{\,2}$	Dixon $[-1.5, 1.5]^2$
$\cos(x_1)\cos(x_2)\exp\left(-(x_1-\pi)^2+(x_2-\pi)^2\right)$	Easom $[-10, 10]^2$
$-\frac{x_1^{\,2}+x_2^{\,2}}{4000}+\cos(x_1)\cos\left(\frac{1}{\sqrt{2}}x_2\right)+3$	Griewangk $[-50, 50]^2$
$x_1\sin\left(\sqrt{\lvert x_1\rvert}\right)+x_2\sin\left(\sqrt{\lvert x_2\rvert}\right)$	Schwefel $[-500,500]^2$

Fig. 6 shows the test functions and Table I the formulas and domains of the functions. Each algorithm was run on each objective function 500 times. We are always looking for maxima of the objective functions. All tested objective functions have two dimensional solutions. Each algorithm is tested with several parameters.

As "bad anchor" random sampling (RND) of the functions while remembering the best function value was added as "optimization algorithm".

We tested the number of function evaluations to reach the 95% mark of the global maximum. In Fig. 7 the mean number of function invocations of all repetitions is shown.

Fig. 8 shows the best function value that is reached after 500 function evaluations. In this figure the median of the function values of all of experiments is shown as a percentage of the global maximum. The figure also shows the 0.975 and 0.025 quantiles. Thus, in 95% of all cases the function value is inside the shown interval. We choose the median and quantiles instead of mean and two standard deviations, because the data is not normally distributed.

In both figures (Fig. 7, 8) the results of the best parameters for each algorithm is shown.

In all experiments (except Random Sampling on Easom and Simulated Annealing on Schwefel) the 95% mark was reached (much) faster than 500 function evaluations. Thus, the results illustrated in Fig. 7 are important for applications in which a fast approximate solution is sufficient, and the results in Fig. 8 are more important if an accurate solution is needed.

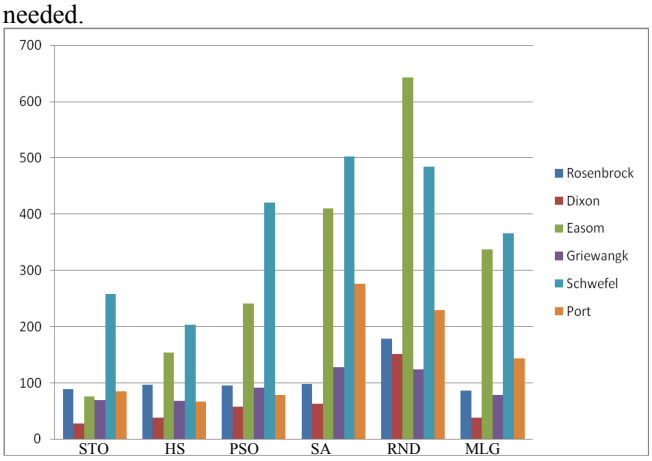


Figure 7. Mean convergence speed of Seed Throwing Optimization (STO), Harmony Search (HS), Particle Swarm Optimization (PSO) . Simulated Annealing (SA), Random Sampling (RND) and Multi-Level Gradient Ascent (MLG). Ordinate: Number of function evaluations to reach 95% mark of global maximum. Benchmark repeated 500 times.

STO was tested with best memory size three and five. There was no significant difference. In all test functions except for Rosenbrock´s function the results were better with $n_{MaxSeeds}$ = 5 than with $n_{MaxSeeds}$ = 10 or $n_{MaxSeeds}$ = 15. In the case of Rosenbrock´s function the results were achieved by choosing $n_{MaxSeeds}$ = 15.

We tested STO without gradient information and were only choosing random directions for "thrown out" seeds. In Rosenbrock´s and in Griewangk´s function the results were better. The "sharp edges" in these functions make it hard to

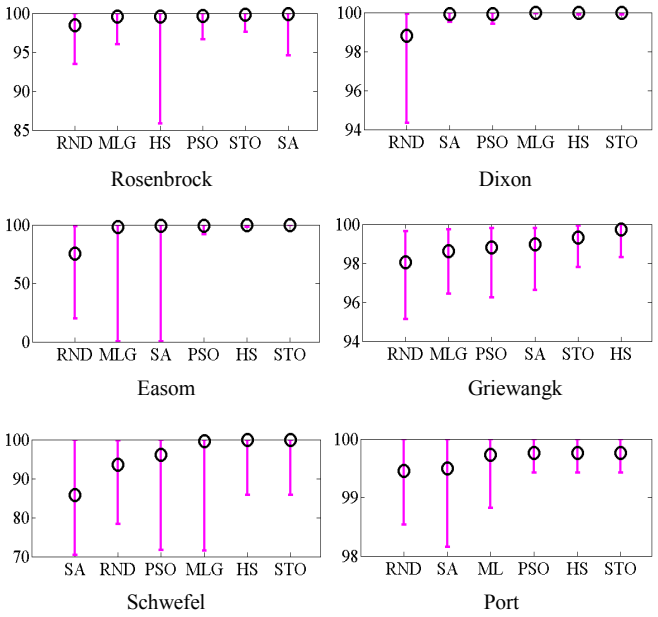


Figure 8. Function value reached after 500 function examinations as percentage of global maximum. Six test functions. Ordinate: median, 0.975 and 0.025 quantiles. The algorithms are sorted by their performance on the test functions.

approximate the steepest ascent. Each approximation of the gradient leads to two further evaluations. Thus, there is a trade-off between a focused search direction and the costs of computing the search direction. As expected, all other test functions showed better results using the gradient.

In another test, the inverse of the second derivation was used to control the step width of "thrown out" seeds like in the Gauss-Newton algorithm [3], but the result was less speed of convergence for most test functions, because two further function evaluations are needed for estimating the second deviation. With a synchronous parallel version all function evaluations for the derivations can be computed in parallel; thus in this case this option is appropriate to achieve a higher speed of convergence.

PSO was tested with 20, 30, 40, 50, 75 and 100 particles. For reaching 95% of the optimum, best results were achieved with 30 to 50 particles, except for Schwefel´s function (100 particles). After 500 function evaluations best results were achieved with 75 to 100 particles. This result is not surprising because the particles are initialized by random sampling and reaching the 95% mark was achieved by 57 to 95 function evaluations for all test functions except Easom and Schwefel.

Harmony search was tested with the best memory size of 5, 10 and 15. In most test functions B_{size}=5 showed best results.

In *Simulated Annealing* the exponential temper plan $T_0 c^t$ with c=0.8, initial temperature T_0=1 and maximum number of iterations of 1000 showed best results for most test functions.

Multi-Level Gradient Ascent was tested with several step widths and several maximum numbers of iterations for the vanilla Gradient Ascent. The results strongly depend on a right choice of the step width.

VI. CONCLUSION

After 500 function evaluations (Fig. 8) best results for Dixon´s, Easom´s and Schwefel´s function are achieved by Seed Throwing Optimization, the best result for Griewangk´s function is achieved by Harmony Search and the best results for Rosenbrock´s function by Simulated Annealing. Particle Swarm Optimization, Harmony Search and STO perform similarly for the port optimization application.

The test functions could be categorized into functions which are highly correlated and into functions where there is a low data-dependency in a local neighborhood. In functions with high correlation the gradient indicates the direction of the global maximum well. This is the case for Rosenbrock´s function, Dixon´s function and for the Port-function, whereas Easom´s, Griewangk´s and Schwefel´s function are more uncorrelated. We can conclude that Seed Throwing Optimization und Harmony Search performs best for both kinds of functions and are the best choices for the tested set of objective functions.

Another conclusion is that Particle Swarm Optimization, Multi-Level Gradient Ascent and Simulated Annealing perform better for the uncorrelated test functions than for the correlated ones.

In Simulated Annealing it was most difficult to find parameters which work well for all test functions.

Seed Throwing Optimization is suitable for synchronous parallelization but excels in asynchronous parallelization. Harmony Search and Particle Swarm Optimization are also well suitable for parallelization. Simulated Annealing and Gradient Ascent do not seem to be suitable for maximum performance in parallelization, since they contain strong dependencies in each iteration of the main loop.

The results (Fig. 7, Fig. 8) show that especially Seed Throwing Optimization and Harmony Search perform very well for a wide variety of functions. The solutions are highly reliable (0.025 quantile).

Moore´s law cannot be continued indefinitely. Hence parallel computing becomes more and more important. Many future computational applications in different domains comprise optimization problems. The asynchronously parallelized version of Seed Throwing Optimization is capable of solving optimization problems without insight of the objective function, it is easy to configure and very computationally efficient. Thus, future computational applications in many domains can benefit from using Seed Throwing Optimization.

ACKNOWLEDGMENT

The present study was conducted within the setting of the "Research training group 1126: Intelligent Surgery - Development of new computer-based methods for the future workspace in surgery" funded by the German Research Foundation.

REFERENCES

[1] C. Blum and A. Roli, "Metaheuristics in Combinatorial Optimization: Overview", ACM Comput. Surv. 35, 268-308 (2003).

[2] T. Weise, "Global Optimization Algorithms – Theory and Application", http://www.it-weise.de/projects/book.pdf [5-5- 2011].

[3] O. Weede, A. Kettler and H. Wörn, "Seed Throwing Optimization: A Probabilistic Technique for Multimodal Function Optimization," 2009 Computation World, pp. 515-519, 2009.

[4] Internet: http://www.intuitivesurgical.com/ [5-5-2011]

[5] Z.W. Geem, J.H. Kim and G.V. Loganathan, "A new heuristic optimization algorithm: Harmony search," Simulation 76, 60-68, 2001.

[6] J. Kennedy and R. Eberhart, "Particle Swarm Optimization", Proc. IEEE Int. Conf. on Neural Networks, 1995.

[7] S. Kirkpatrick, C.D. Gelatt and M.P. Vecchi, "Optimization by Simulated Annealing", Science, New Series, Vol. 220, No. 4598, pp. 671-680, 1983.

[7] D. R. Greening, "Parallel Simulated Annealing Techniques", Physica D: Nonlinear Phenomena, Elsevier, pp.293-306, 1990

[8] J.F. Schutte, J.A. Reinbolt, B.J. Fregly, R.T. Haftka and A.D. George, "Parallel global optimization with the particle swarm algorithm", Int. J for Numerical Methods in Engineering, pp.61:2296-2315, 2004

[9] C. Jay and R. Hubbold, "Delayed Visual and Haptic Feedback in a Reciprocal Tapping Task", Proc. of the First Joint Eurohaptics Conf. and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, IEEE, pp. 655 – 656, 2005

[10] H. Pohlheim, "GEATbx: Example Functions 2 Parametric Optimization", Internet: http://www.geatbx.com/docu/fcnindex-01.html#P160_7291 [5-5-2011]

[11] G. Riley, "Network Simulation with ns-3", Spring Simulation Conference, April 12, 2010

[12] H. Wörn and O. Weede, "Optimizing the setup configuration for manual and robotic assisted minimally invasive surgery," in World Congress on Medical Physics and Biomedical Engineering 2009, Munich, Germany, 2009, pp. 55—58, 2009

 ISBN: 978-1-61208-154-0

A Case Study on the Design Trade-off of a Thread Level Data Flow based Many-core Architecture

Zhibin Yu, Andrea Righi, Roberto Giorgi
Department of Information Engineering
University of Siena (UNISI)
Via Roma, 53100, Siena, Italy

***Abstract*— With the potential of overcoming the memory and power wall, the many-core/multi-thread has become a trend in processor design area. However, this architecture is far from ripeness because it also companies with many challenges such as scalability and larger architecture design space compared with mono-core architectures. In many-core design space, Data-Flow based architectures are alternatives that deal with concurrency, long memory latencies, and synchronization stalls efficiently. Nevertheless, even in this sub-area, there are still a lot of factors affecting the scalability and performance of the architecture. In this paper, we explore the design trade-offs for Decoupled Threaded Architecture (DTA) which is a data-flow many-core architecture. By using a well known bio-informatics benchmark, ClustalW, we evaluate various DTA configurations with different number of synchronization and execution pipelines. We find that the configuration which consists of two synchronization pipelines (SP) and one execution pipeline (EP) for each processing element(PE) achieves almost the same performance as the configuration consisting of two SPs and two EPs for each processing element. By employing the former configuration, we can save 32.5% of the area required for each DTA processing element.**

Keywords-Multi-threaded;Many-core;Scalability;Programability; Dataflow architecture.

I. INTRODUCTION

Due to the limited performance gains from mono-core architectures, the industry has already shifted gears to deploy architectures with multiple cores and threads [22]. Although multi/many-core architectures promise a significant performance potential, it is not trivial to obtain performance improvement from these architectures. Besides traditional difficulties, new challenges such as scalability and programmability are arising. Further, the design space of multi/many-core architectures is much larger than that of mono-core architectures, leading more difficulties to design them. The Data-Flow architecture [2] is an alternative that deals with concurrency, long memory latencies, and synchronization stalls efficiently. We have designed a multi-threaded architecture named Decoupled Threaded Architecture (DTA) based on Data-flow architecture [1].

Decoupled Threaded Architecture (DTA) is a multi-threaded architecture based on the Scheduled Data Flow (SDF) execution paradigm. The way in which data is communicated among threads and the decoupling of memory accesses from execution are the main differences between DTA/SDF and other multithreaded programming models. Data is exchanged between threads via frames which are portions of local memory assigned to each thread. Each thread is associated with a Synchronization Counter (SC) that represents the number of input data needed by the thread. This counter decreases each time when a datum arrives in the thread's frame. Once it becomes zero, which means all needed input data are ready, the thread is ready to execute. In this way, DTA provides dataflow at thread level and a non-blocking synchronization. This is one of the key differences between the DTA and the original Data-flow architecture which provides dataflow at instruction level.

In this paper, we provide a case study for the design trade-off our previously proposed DTA architecture. We employ the well-known bio-informatics application Clustal-W to evaluate various DTA configurations. Especially, we look for the optimal combinations of the number of synchronization and execution pipelines. We show that the configuration of 2 SPs and 1 EP achieves the same performance as that of 2 SPs and 2 EPs. Therefore, we can save 32.5% of the area required for each DTA processing element by using the former configuration.

The rest of this paper is organized as follows. Section II briefly describes an overview of the DTA architecture. Section III provides an analysis of the benchmark Clustal-W. The experimental methodology is given in Section IV and Section V provides the results and analysis. Section VI surveys the related work and we conclude the paper in Section VII.

II. OVERVIEW OF THE DTA ARCHITECTURE

A. The Execution Model

DTA executes TLP (Thread Level Parallelism) activities of a program — portions of a program that exhibits Thread Level Parallelism (TLP). A Compiler (or a programmer) identifies parallel parts of the program and marks them as TLP activities. When these activities are encountered during execution, they are launched to the DTA hardware where they are executed in parallel. For example, in the DTA implementation on the Cell processor [3], TLP activities are launched by the general purpose processor (Power PC) to the DTA-enabled Synergistic Processor Elements that execute DTA threads.

The DTA architecture decouples the memory accesses from execution. This helps the threads exchange data in the data flow manner. To achieve this, a new memory concept, frames which are portions of local memory assigned to each thread, is introduced. One thread has one frame which is used

 ISBN: 978-1-61208-154-0

to store the input data for the thread. In order to indicate whether all the input data needed by a thread is in the corresponding frame, the DTA architecture employs a Synchronization Counter (SC). This counter represents the number of input data needed by a thread. When one datum of a thread arrives at its frame, the SC of it decreases by one. Once a SC becomes zero, the corresponding thread is ready to execute. The execution of each DTA thread can be split into three phases: load, execution, and store. In the load phase, input data are loaded from the frame memory; in execution phase, the thread is executed; in the store phase, the outputs of the thread are written to the frames of other threads. Preemptive execution is not allowed and a thread voluntarily releases the processor each time when it switches between phases. Therefore, the memory access of a thread is decoupled from the execution of it.

An example of thread synchronization in DTA is shown in Figure 1. Thread *th0* executes first and creates threads *th1*, *th2* and *th3*. Since the thread *th1* needs two input variables *a* and *b*, its SC is set to 2 when the thread is created. Similarly, the SCs of threads *th2* and *th3* are set to 1 and 2 respectively at the beginning. When *th0* executes the first STORE instruction (used to send data to another thread), it will store *a* into the frame of the thread *th1*. Meanwhile, the SC of *th1* is decreased by 1. After the second STORE, the SC of *th1* becomes 0 and *th1* is ready for its execution. When the third STORE of *th0* completes, *th2* is also ready and both threads (*th1* and *th2*) can run in parallel if there are two cores available. When their execution completes, the output data are stored into the frame of thread *th3*.

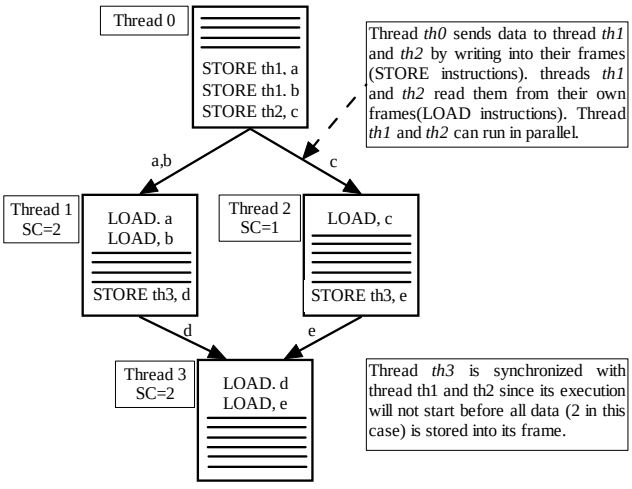


Figure 1. An example of thread synchronization

To overcome the long wire delay issue [4], the DTA architecture clusters resources into nodes, a different approach from the SDF architecture. Figure 2 shows the overview of our DTA architecture. As shown in Figure 2, each node contains several processing elements (PE) that are interconnected via a fast and simple intra-node network. Each node is dimensioned so that all PEs in a single node can be synchronized by using the same clock. The Nodes are connected by a slower inter-cluster network.

B. *The DTA Architecture*

The DTA architecture employs two kinds of schedulers to schedule workloads among the computing elements: Distributed Scheduler Elements (DSEs) and Local Scheduler Elements (LSEs), which are illustrated in Figure 2. Each PE contains one LSE that manages local frames and forwards request for resources to the DSE. For example, when a remote store arrives at a local frame of a thread, the LSE decreases the SC of the thread and stores the datum to the frame. When a PE requests a new frame, the request is forwarded to the DSE. Each node contains one DSE that balances the workloads among processors in the node. The elements of the schedulers communicate with each other by using messages [1].

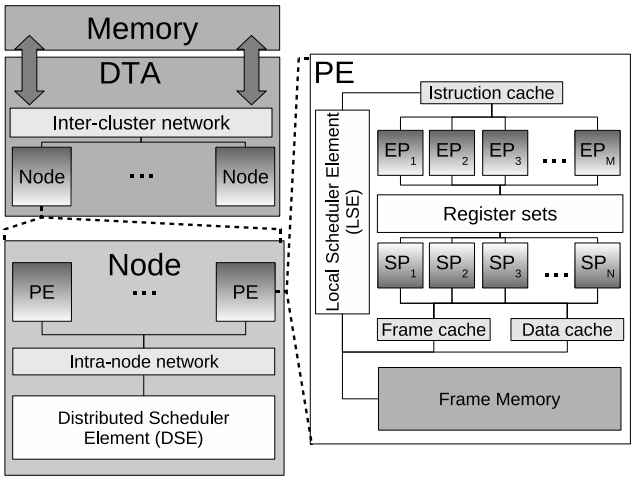


Figure 2. DTA architecture design. A processing element (PE) is composed of many execution pipelines (EP) and synchronization pipelines (SP) with common register sets, frame, data, and instruction cache. Processing elements are grouped by nodes to break scalability limits.

There are two types of memory in the DTA architecture:

- **Frame Memory (FM)**. It is private for each thread and contains input data for the thread;
- **Global Memory (GM)**. It is shared among threads and is accessible via the inter-cluster network;

Both frame memory and global memory use a direct- mapped cache of 32KB (no additional overhead is considered in case of cache hits) to reduce the memory latency. Reading from frame memory is always faster than reading from global memory. On the other hand, writing data to both types of memory is always non-blocking since the LSEs take the write request and process it. The PE is free to continue executing as soon as possible after the request is passed to a LSE. In order to reduce the memory latency, DTA programs tend to use the frame memory as much as possible.

C. *The Processing Element*

The processing elements in DTA can be either off-the-shelf processors or specialized DTA processors with separate pipelines for different phases of each thread. When off-the-shelf processors are used, they need to be modified in order to include LSE, frame memory (usually cache or local store is

used [3]) and several DTA-specific instructions which are used to manage the lifetime of threads and exchange data among threads. In order to extract precise statistic information for each phase of a thread's execution, we use DTA-specific processing elements in this study.

Each PE contains several pipelines, a register set, a frame memory, and a Local Scheduler Element. There are two types of pipelines in each PE: Synchronization Pipeline (SP) and execution pipeline (EP). SP is responsible for executing load and store phases. EP is responsible for the execution phase of a thread. All pipelines in the PE share the same register set. When a thread starts to execute, the LSE assigns one of the available register sets to it. Then the thread passes trough different pipelines to execute its load, execution and store phases. When the thread finishes its execution, its frame and register set become available for other threads.

All requests that originate from a PE (either for memory accesses or for new resources) are handled by its LSE. If the request is local (mapped to internal frame memory), then it is served immediately. On the other hand, in case of a request for remote resource (new frame or a remote memory location), the LSE forwards the request to the DSE (in case of a request for a new frame or a store to a remote frame) or to the main memory (in case of a request for a remote memory location). In the both cases, the request must pass through intra-cluster network, causing a longer latency.

D. Terminologies

In this paper we use the following terminologies to differentiate the types of threads:

- **Pthread**: a thread created by an application [5]. These threads are specified by the programmer and can have arbitrary length.
- **DTA-thread**: a sequence of load, execution and store phases. Each pthread contains many small DTA threads that are generated by the compiler.

III. The Analysis of Clustal-W

In molecular biology, Clustal-W [6] is an important program for the simultaneous alignment of nucleotide or amino acid sequences. It implements the most widely used approach of multiple sequence alignments that use a heuristic search known as progressive alignment. However, the progressive alignment algorithm suffers a high computational complexity and consequently it may take a lot of time to complete. A traditional technique to speedup this task is to parallelize the application as much as possible. As a result, the progressive alignment algorithm is a perfect candidate to fully utilize the machine resources and to stressfully test the overall performance as well as the scalability of massive parallel systems.

In this work, we use the parallel implementation of Clustal-W that comes from the BioPerf [7] benchmark suite. This version of Clustal-W is divided in three phases:

1) The first phase, pair-wise alignment, takes between 60% and 80% of the execution time in a traditional uni-processor machine.
2) The second stage forms a phylogenetic tree using the Neighbor-Joining algorithm with the aligned sequences generated at the previous step.
3) The third step progressively aligns the sequences according to the tree branching order obtained at the second step.

In order to better understand the performance of Clustal-W, we characterize it by using Oprofile [8]:

Counted CPU_CLK_UNHALTED events

samples	%	image name	symbol name
526230	50.0853	clustalw-smp	parallel
271232	25.8152	clustalw-smp	pdiff_reverse
215633	20.5234	clustalw-smp	pdiff_forward
15849	1.5085	clustalw-smp	pdiff
14632	1.3926	clustalw-smp	calc_prf1
2535	0.2413	clustalw-smp	diff
1731	0.1648	clustalw-smp	prfalign
1534	0.1460	clustalw-smp	aln_score

The above profiling results show that the pairwise alignment — function *parallel()* — is the most CPU-consuming part of the execution. Over 50% of execution time spent in it. Hence, we focused our analysis on this function.

IV. Experimental Methodology

A. The Methodology

As a starting point, we use the Clustal-W implementation that is parallelized at the application level by using Pthreads primitives. In order to better utilize the underlying architecture, the pthread library was implemented by using DTA assembly primitives. Whenever a new pthread is requested in the code, the compiler creates a DTA thread to exploit the dataflow execution model. Instead of relying on locking and semaphores, pthread primitives for synchronization are written to rely on the synchronization counters and the dataflow communications that are supported by the DTA hardware.

The standard ANSI-C version of Clustal-W is translated into DTA assembly code by using a modified Scale [18] compiler. This compiler is extended with a DTA backend and custom implementation of libraries for the generic I/O operations (*open()*, *read()*, *write()*, . . .). The tests are executed on a DTA simulator which is based on the code of sdfsim-3.0.0 [19].

For the input dataset, we use the standard input dataset from the BioPerf suite, 1290.seq (66 sequences of length almost 1100). The input is replicated 32 times to create at least 512 workers via *pthread_create()* and to avoid "empty" worker threads. All the created threads have a non-empty input sequence to align. The total number of sequences to be aligned during each run is 66 x 32 = 2112.

 ISBN: 978-1-61208-154-0

B. *The Decomposition of the code*

Since the main goal of this work is to better understand the TLP exploitation from the point of view of the architecture, we focused on the statistics for the highlighted section of the code (see Section III). This section starts after all threads are created. Namely, it starts after the call to the function *pthread_cond_broadcast()*.

As mentioned before, the main idea is to allow the DTA to coexist with other types of processors in the system (e.g. general purpose processors) and to launch only the parallel part of the code to a dedicated DTA hardware. In this way, the General Purpose Processor (GPP) executes only for the sequential part of the application. The GPP can be optimized to exploit the available Instruction Level Parallelism (ILP) of the program that is not suitable for DTA. On the other hand, the TLP-optimized DTA hardware runs the threaded portion of the code. By analyzing the code and the arguments of *pthread_create()* function, it is fairly easy to identify the portion of the code that has the high degree of parallelism. Therefore, even at compile time, we can decide how to split the code into sequential and parallel parts.

V. RESULTS AND ANALYSIS

Firstly, we estimate the instruction mix that is created at runtime:

Total number of instructions:	2,323,534,075
Total frame-memory references:	1,097,224,437
Total data-memory reference:	91,925,906
Total READs from frame:	548,612,219
Total WRITEs to frame:	548,612,218
Total READs from memory:	78,648,743
Total WRITEs to memory:	13,277,163
Total number of DTA-threads created:	50,259,676
Instructions per DTA-thread (average):	47.2305

As we can see from the dynamic statistics, 51.12% of the instructions are memory accesses but only 3.96% of them are the data memory access. The 96.04% of memory accesses goes to the frame memory and these accesses are used for the communication among DTA-threads. This is evidence that the compiler is able to generate many smaller DTA threads inside each thread created by pthread library. Since the frame memory locates near the processor, the advantage of DTA is to leverage this memory for communication among threads instead of using other ways of communication (e.g. function calls with a stack frame).

The second set of experiments test the scalability of several different configurations of DTA while varying the miss latency. The results are shown in Figure 4 and Figure 5. In the DTA architecture, the stalls in the pipelines are only a consequence of LOAD misses. These stalls delay the fetch and completion of future instructions. Therefore, only the Synchronization Pipelines are affected by pipeline stalls. Increasing the memory latency reduces the contribution of the EPs to the total execution time. On the other hand, the activity of SPs strongly influences the performance in terms of total execution time and speedup. This can be seen from the graphs for execution time since the configuration with 2 SPs and 1 EP performs almost as well as the configuration with 2 SPs and 2 EPs. Also, increasing the memory latency gives better speedup (but the execution time increases also) since there is a higher probability to use more SPs.

When the latency is increased, the amount of stalls in SPs, as shown in Figure 6, is also increased so do the number of threads in the system as illustrated in Figure 7. The number of threads is increased because there are many new threads are waiting. If we have a longer stall in a pipeline and a new DTA-thread arrives, there is a less probability to find an available SP that can start to execute the load phase of the thread. This means that the benefit of increasing the number of SPs is greater than increasing the number of EPs when the memory latency increases.

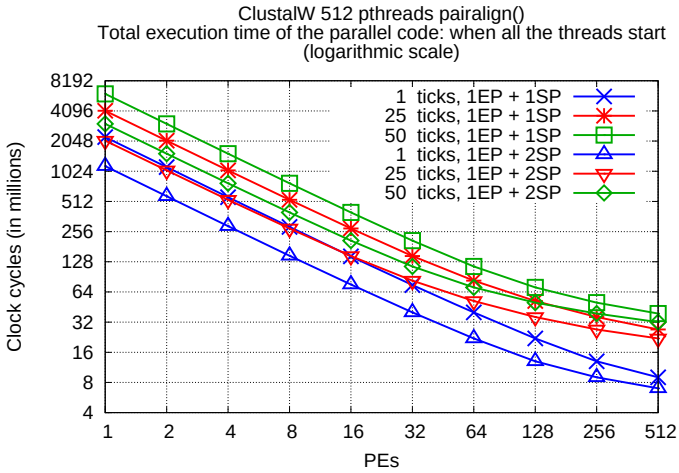


Figure 4 Total execution cycles with variant PE configurations and Load miss latency. Memory latencies are 1, 25, and 50 clock cycles (ticks)

This fact is due to the memory latency issue and execution model. Statistically, at any time of the execution of a program, 2/3 of the available DTA-threads need to be served by a SP (due to load and store phases) and only 1/3 of the threads need an EP to continue their execution. This is also in line with the obtained results on execution time and speedup that are discussed above. Hence, in a dataflow base multithreaded architecture, it is possible to get the same performance by using a cheaper configuration, namely, by decreasing the number of architectural elements that are dedicated to calculations (EPs) and by increasing the number of architectural elements dedicated to communication (SPs).

To quantify the area saved by decreasing the number of EPs, we evaluated the number of transistors and the required area for all configurations that we used in our experiments. Our method is based on the analytical method provided by the "SimpleScalar directed estimation tool" [20]. This tool takes the number of functional units (such as ALUs, Load/Store units,...) as input and produces the estimate of required transistor count and the area. The results are displayed in Table 1. As shown in Table 1, the "1 EP + 2 SPs" configuration saves about 32.5% of the eara with respect to the "2 EPs + 2 SPs" configuration while keeps the same

 ISBN: 978-1-61208-154-0

performance, leading to a space and energy efficient architecture.

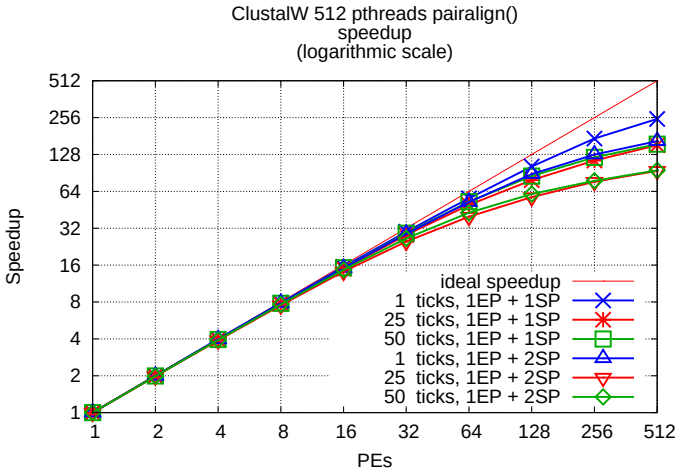


Figure 5. Speedup with different PE configurations and LOAD miss latency. The memory latencies are 1, 25, and 50 clock cycles (tickets).

VI. RELATED WORK

Several researchers in the past have studied the performance of multiple versions of parallel Clustal-W in a wide range of different multi-core architectures. The solution presented by [9] uses message-passing libraries on a PC cluster (ClustalW-MPI). Besides the software approach, new approaches that are using reconfigurable hardware (such as FPGA) have been presented [10]. Vandierendonck et al. [11] explored the performance of a Clustal-W implementation optimized for the Cell BE architecture (ClustaIW-Cell). Liu et al. also explored the optimizations of Clustal-W using the GPU acceleration of nvidia GeForce 7800 GTX (ClustalW-GPU) [12].

From the hardware perspective many decoupled architectures have appeared in the past few years. Speculative Data- Driven Multithreading [13] is an architecture that is based on decoupling principle. This architecture identifies miss streams, i.e. streams of instructions that are likely to cause cache misses and executes them in a multithreaded fashion in order to perform pre-fetching. HiDisc (Hierarchical Decoupled Instruction Stream Computer) [14] is an architecture that reduces memory latency by pre-fetching at both hardware and software level. Pre-fetching is accomplished by separating the instruction stream into one for regular execution and one for memory accesses.

Moreover, multi-core/many-core architectures have gained the most attention in the industry recently. IBM Cyclops-64 (C64) [15] is a multi-core-on-a-chip processor that consists of 80 processors (or cores). Each processor has two SRAM memory banks that can be configured either as scratchpad or global memory. Plurality [16] is a multi-core system that uses a pool of RISC processors with uniform memory, hardware scheduler, synchronizer and load balancer. SUN Microsystems UItraSPARC T2 [17] is a multithreading multi-core chip capable of running 64 threads at the same time. The main difference between the existing architectures and DTA is that DTA is a multithreaded architecture that uses the scheduled dataflow programming model and decouples

TABLE I
AREA AND TRANSISTOR USAGE ESTIMATION:
PIPELINES AND DIFFERENT PROCESSING ELEMENTS

Element	Number of Transistors	Area (in M λ^2)
1 EP	419,597	685.47
1 SP	225,554	368.49
1 PE = 2 EP + 2 SP	1,290,302	2,107.92
1 PE = 1 EP + 2 SP	870,705	1,422.45

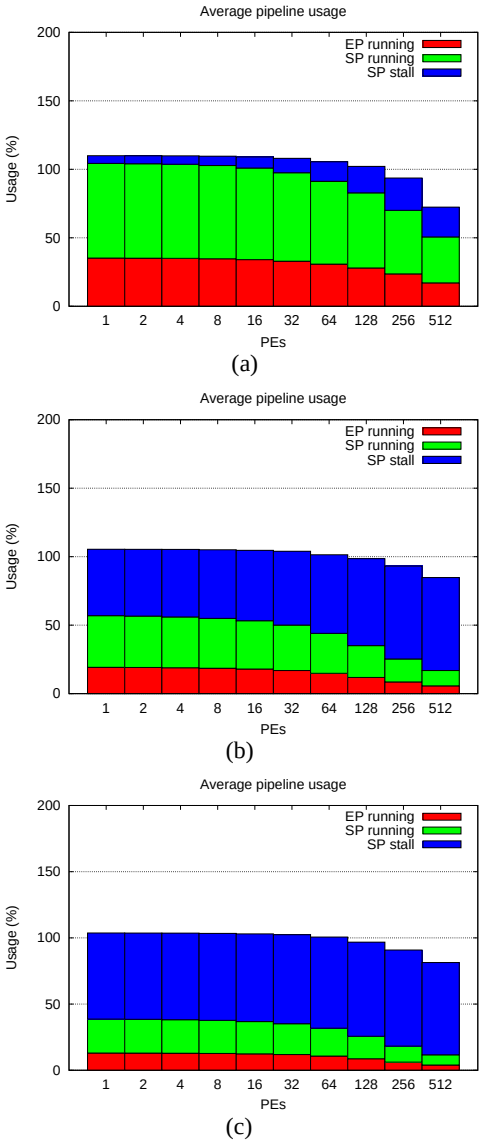


Figure 6. Average pipeline usage (1PE = 1 EP + 1 SP) with a Load miss latency of 1, 25, and 50 clock cycles, denoted by (a), (b), and (c). Total usage can be greater than 100% when EP and SP code are executed in parallel

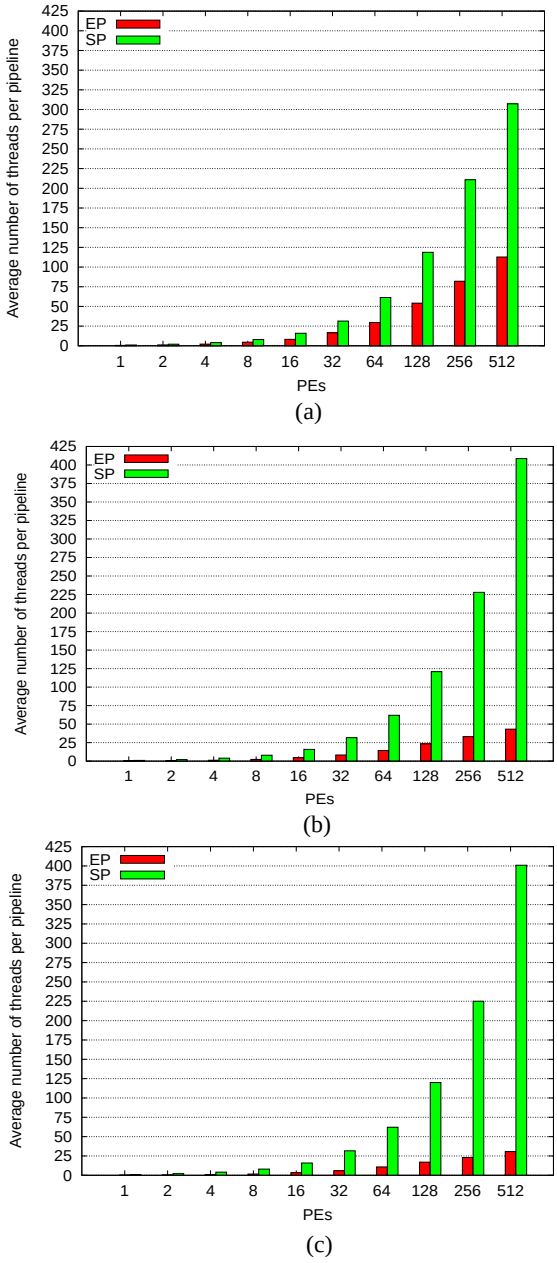


Figure 7. Average number of DTA-threads running in EPs and SPs using different LOAD miss latencies: 1, 25, 50 clock cycles(ticks)

the memory accesses from the threads' execution.

VII. CONCLUSION

This paper analyzes the tradeoffs in design of a multi-threaded architecture, and their effect on the exploitation of the Thread Level Parallelism. By using the bio-informatics application Clustal-W as a benchmark, we evaluate our Decoupled Threaded Architecture (DTA) with different number of architectural elements that are dedicated for computation and for communication among threads. We have experimentally verified that the contribution of the hardware dedicated to communication (Synchronization Pipelines — SPs) is much greater than the contribution of the hardware dedicated to computation (Execution Pipelines — EPs). This shows that our architecture is based on the coarse-grained dataflow execution model that emphasizes the communication in a producer-consumer fashion. We found that the configuration in which each Processing Element (PE) is composed of 1 EP and 2 SPs is able to achieve the performance which is very close to that of the configuration with "2EP + 2SP". This can save the area of each PE and obtain benefits in terms of power significantly.

ACKNOWLEDGEMENTS

We thank Prof. Krishna Kavi for providing the modified Scale compiler and for his useful comments. This work was partly funded by the European FP7 project TERAFLUX id. 249013[21], HiPEAC IST-217068, and IT PRIN 2008 (200855LRP2).

REFERENCES

[1] R. Giorgi, Z. Popovic, and N. Puzovic, "Dta-c: A decoupled multi-threaded architecture fhr cmp systems," in *Proceedings of IEEE SBAC-PAD*, Gramado, Brasil, Oct. 2007, pp. 263-270. [Online]. Available: http://www.dii.unisi.it/ popovic/docs/Giorgi-DTA.pdf

[2] K. M. Kavi, R. Giorgi, and J. Arul, "Scheduled dataflow: Execution paradigm, architecture, and performance evaluation," *IEEE TRANSACTIONS ON COMPUTERS*, pp. 834-846, 2001.

[3] R. Giorgi, Z. Popovic, and N. Puzovic, "Introducing hardware TLP support or the Cell processor," in *Proceedings of IEEE International Workshop on Multi-Core Computing Systems. Fukuoka, Japan*. Los Alamitos, CA, USA: IEEE Computer Society, Mar 2009, pp. 657-662.

[4] R. Ho, K. Mai, and M. Horowitz, "The future of wires," *Proceedings of the IEEE*, vol. 89, no. 4, pp. 490-504, Apr 2001.

[5] B. Nichols, D. Buttlar, and J. P. Farrell, "Pthreads programming: A POSIX standard for better multiprocessing," *Reilly, California*, 1996.

[6] J. D. Thompson, D. G. higgins, and T. J. Gibson, "CLUSTAL w: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice," *NUCLEIC ACIDS RESEARCH*, vol. 22, pp. 4673-4673, 1994.

[7] D. A. Bader, Y. Li, T. Li, and V. Sachdeva, "BioPerf: a benchmark suite to evaluate high-performance computer architecture on bioinformatics applications," in *Proceedings of the IEEE International Workload Characterization Symposium*. 2005, pp. 163-173.

[8] J. Levon and P. Elie, "Oprofile: A system profiler for linux," Web site: *http://oprofile. sourceforge. net* , 2005.

[9] K. B. Li, "ClustalW-MPI: ClustalW analysis using distributed and parallel computing," *Oxford Univ Press*, 2003, vol. 19.

[10] T. Oliver, B. Schmidt, D. Nathan, R. Clemens, and D. Maskell, "Using reconfigurable hardware to accelerate multiple sequence alignment with ClustalW," *Oxford Univ Press*, 2005, vol. 21.

[11] H. Vandierendonck, S. Rul, M. Questier, and K. D. Bosschere, "Experiences with parallelizing a bio-informatics program on the cell BE," *LECTURE NOTES IN COMPUTER SCIENCE*, vol. 4917, p. 161, 2008.

[12] W. Liu, B. Schmidt, G. Voss, and W. Muller-Wittig, "GPU-ClustalW: using graphics hardware to accelerate multiple sequence alignment," *LECTURE NOTES IN COMPUTER SCIENCE*, vol. 4297, p. 363, 2006.

[13] A. Roth and G. S. Sohi, "Speculative Data-Driven multithreading," in *Proceedings of the 7th International Symposium on High-Performance Computer Architecture*, vol. 37, 2001.

[14] W. W. Ro, S. P. Crago, A. M. Despain, and J. L. Gaudiot, "Design and evaluation of a hierarchical decoupled architecture," The *Journal of Supercomputing*, vol. 38, no. 3, pp. 237-259, 2006.

 ISBN: 978-1-61208-154-0

[15] G. Almási, C. Cacaval, J. G. Castaños, M. Denneau, D. Lieher, J. E. Moreira, and II. S. Warren, Jr., "Dissecting cyclops: a detailed analysis of a multithreaded architecture," *SIGARCH Comput. Archit. News*, vol. 31, no. 1, pp. 26-38, 2003.

[16] "Plurality architecture." [Online]. Available: http://www.plurality.com/architecture.html. June 29, 2011.

[17] M. Shah, J. Barreh, T. Brooks, R. Golla, G. Grohoski, N. Gura, R. Hetherington, P. Jordan, M. Luttrell, C. Olson, *et al*., "UItraSPARC T2: A highly-treaded, power-efficient, SPARC SOC," in *Solid-State Circuits Conference, 2007. ASSCC'07. IEEE Asian*, Jeju, Republic of Korea, 2007, pp. 22-25.

[18] K. S. McKinley, J. Burrill, M. D. Bond, D. Burger, B. Cahoon, J. Gibson, J. E. B. Moss, A. Smith, Z. Wang, and C. Weem, "The Scale compiler," *Technical report*, University of Massachusetts, 2001. http://ali-www. cs. umass.-edu/scale, 2005.

[19] SDFsim 3.0.0, http://csrl.unt.edu/sdf/sdfhowto.php June 29, 2011.

[20] M. Steinhaus, R. Kolla, J. L. Larriba-Pey, T. Ungernr, and M. Valero, "Transistor count and Chip-Space estimation of simulated microprocessors," *Research report UPC-DAC-2001-16*, UPC Barcelona Spain, 2001.

[21] http://www.Teraflux.eu June 29, 2011.

[22] S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, N. P. Jouppi, "McPAT: An Integrated Power, Area, and Timing Modeling Framework for Multicore and Manycore Architectures," in *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture*, IEEE Computer Society, Dec 2009, Notre Dame, IN, USA, pp. 469-480.

 ISBN: 978-1-61208-154-0

Theoretical Analysis and Simulation to Investigate the Fundamental Rules of Trust Signaling Games in Network-based Transactions

So Young Kim, Junseok Hwang
Technology Management, Economics and Policy Program
Seoul National University
Seoul, Republic of Korea
none8171@snu.ac.kr, junhwang@snu.ac.kr

***Abstract*—Online network-based transactions are widespread forms of transactions in e-commerce markets such as peer-to-peer markets or smart media markets. In these markets, the participants need criteria to search, select and manage their partners. One of the most important criteria is the trustworthiness of the partner. The participants aim to enhance the probability of being selected by their opponents through signaling their trustworthiness levels to their opponents. Simultaneously, the opponents adjust their beliefs on the trustworthiness of other participants based on observation of signals. This paper describes this situation using a signaling game in which the seller sends a signal of his/her trust level and the buyer decides his/her payment schedule for the presented signal. The results of the equilibrium analyses suggest criteria for the signaling of the cost structures of participants and the market environment. Additionally, the results of the simulations validate the results of the equilibrium analyses.**

Keywords-trust; signaling game; equilibrium; agent-based simulation

I. INTRODUCTION

Autonomous agents in the network-based transaction market need criteria to search for other participants, select partners and manage relationships. One of the most important criteria is the trustworthiness of the partner. Network-based transaction markets such as e-commerce, peer-to-peer markets, business-to-customer markets, and business-to-business markets often only provide information goods [1]. Information goods have the characteristics of experience goods, whose quality is difficult to observe in advance, but can be ascertained with experience [2]. Therefore, information asymmetry is one of the main focuses of many studies that deal with information goods.

Many studies have attempted to mitigate the negative effects of information asymmetry. Researchers have investigated various mechanisms that evaluate the level of trustworthiness of an agent in the network, such as reputation, recommendation and third party authorization [3]. For example, web recommendation systems [4] and trust certification stamp systems [5] are kinds of mechanisms that have been developed to manage trust among a number of unspecified agents in the internet. These previous efforts, however, have focused more on enhancing the accuracy of prediction by developing sophisticated prediction mechanisms. The signal resulting from these mechanisms is also asymmetric, so that an agent has to adjust his/her belief about the trustworthiness of the opponent based on the results of observation.

This paper describes this situation as a signaling game in which the seller (or the sender) sends the signal of his/her trust level and the buyer (or the receiver) decides his/her payment schedule for the presented signal. This paper also presents the criteria of the signaling cost structures of participants and the market environment. Satisfying these criteria ensures the effectiveness of signals in distinguishing each type of participant and the stability of the separating equilibrium. Through these processes, this paper also investigates fundamental rules of trust signaling games in network-based market transactions. Additionally, it also uses an agent-based simulation to validate whether these rules are effective in the market for agents that mimic bounded-rational human behavior.

The remainder of the paper is organized as follows. The next section reviews the related literature. Section 3 proposes the trust signaling model. Section 4 conducts an equilibrium analysis of the trust signaling game. Section 5 validates the theoretical model using an agent-based simulation. Finally, Section 6 provides a discussion and conclusion.

II. LITERATURE REVIEW

Akerlof [6] discussed the problem of information asymmetry and quality uncertainty in the market for used cars. Spence [7] also considered the issue of information asymmetry between employers and employees in his pioneering work. He suggested that the employers use employees' education levels as signals and offer wage schedules on the basis of their beliefs about labor productivity. Spence's model offers a theoretical framework that can describe many kinds of signaling games. This model could be used to describe social relationships based on the trustworthiness signal.

Several recent studies have focused on interactions between autonomous agents in the network using game theory, particularly signaling games. One of these studies applied game theory to detect intrusion by malicious nodes in a mobile ad hoc network without

 ISBN: 978-1-61208-154-0

centralized control [8]. This study provides insights regarding the attacker in the network and the intrusion detection system by modeling the interaction between normal nodes and attackers as a basic Bayesian game. Another study of mobile ad hoc networks focused on the best strategies that normal nodes and malicious nodes can select in a dynamic Bayesian signaling game. It validated the superiority of the suggested strategy and concluded that restricting the opportunity for malicious nodes to flee from detection is important [9].

The mechanisms supporting the decision making about whether one can trust an opponent in network transactions have been considered in various studies over a long period of time. Reputation mechanisms have become a fairly common framework since several pioneering studies [10] [11] and subsequent studies [12]. One subsequent study suggested that the reputation mechanism of the previous opponents of a player offers feedback information about previous transactions; the autonomous system aggregates these feedback with proper weights and then the system offers more accurate evaluation of the opponents' trustworthiness [13].

Another study compared simulation analysis and theoretical analysis. It suggested that while game theory can analyze the behavior of rational agents, agent-based simulations can analyze the behavior of software agents that mimic real-life decision makers [14].

III. MODEL DESCRIPTION

In the simplest signaling game, there are two players—the sender and the receiver—in the set I. The sender can be either malicious type (M type) or normal type (N type). The receiver can only be regular type (R type). The set of types is denoted by T: $T=T_S \times T_R$, $T_S=\{M,N\}$, $T_R=\{R\}$. The type of sender is chosen by nature and is the private information of the sender. Each player has a strategy set A and a utility set u. Therefore, the structure of the game is simply denoted by: $G=\{I,T_i\}_{i\in I},\{P(\cdot|\cdot)\}_{i\in I},\{A_i\}_{i\in I},\{u_i\}_{i\in I}$

The prior probability that the sender is M type or N type is $\pi(Malicioius)=\pi_M$ or $\pi(Normal)=1-\pi_M$. The sender of a particular type sends a message m $(m:T\rightarrow M)$ about his/her level of trustworthiness to a receiver. The message is drawn from the set $M=\{e,0\}$. The receiver receives this signal, and then takes an action drawn from a set A. This action a $(a:M\rightarrow A)$ indicates the value that the receiver is willing to pay to sender, w. The values of w forms the strategy set $A=\{w|w\in R^+\}$. The payoff of player i is given by the function $u_i:T\times M\times A\rightarrow R$. This means that the payoff of a player is decided by the player type, the message of a sender and the action of a receiver.

In our example, M type and N type senders receive the payoffs $v+L$ and v (v, $L\geq 0$) when the transaction is successful. The receiver receives payoffs of $-(v+L)$ if he/she transacts with an M type sender and v if he/she transacts with an N type sender.

The receiver cannot observe the type of transacting sender; therefore, the sender uses a certain form of signal to increase the probability that the receiver chooses him/her as a partner or increase the payoff from a successful transaction. The signal can take various forms such as the disclosure of a transaction history, presentation of a certification from a third party authority, or advertising. Most of these signals involve a cost. For example, if a sender wants to signal by disclosing a transacting history, he/she cannot violate the transaction rules for a given period even if he/she is the M type sender. One can easily imagine that this form of signal costs more for the M type sender than for the N type sender. Some forms of signal may involve an equal cost for the M xtype and the N type senders. However, it is likely that the receiver will be unable to distinguish one type from another if the signal costs of obtaining the same level of trustworthiness for two types of senders are the same. Therefore, we assume that the cost of an M type sender is relatively higher than that of an N type sender.

The sender advertises his/her own type honestly or deceitfully by sending a certain level of message $e\in[0,\infty)$ that appears to represent the trustworthy level. The message e costs $c_M(e)$ for the M type sender and $c_N(e)$ for the N type sender. Then the receiver suggests the fee schedule $w(e)$ that the receiver wants to pay to ensure a trust-based transaction with the sender. Therefore, the expected payoff of the M type sender is $u_M(e)=v+L+w(e)-c_M(e)$ and $u_N(e)=v+w(e)-c_N(e)$ for the N type sender when the transaction is successful.

The potential receivers are assumed to be sufficiently many and to be risk neutral so that they suggest a wage schedule that has the same value of expected profit for a transaction and satisfies the zero profit condition of the competitive market providers [15]. Therefore, the receiver suggests the fee $w(\mu_e)=v-\mu_e(2v+L)$ to ensure a trust-based transaction when he/she has observed the message e from the sender and has the belief μ_e that the sender is an M type. Of course, the receiver does not participate unless $\mu_e < v/(2v+L)$.

IV. EQUILIBRIUM ANALYSIS

A. Separating Equilibrium

Proposition 1. When the level of trustworthiness of a participant is used as the signal, the signal can be effective in distinguishing one sender from another if the cost of trust level signaling is sufficiently distinct.

Proposition 1 indicates that the presented signal gives perfect information of the type of sender if the two sender types select distinct levels of trustworthiness as their signals in the equilibrium. When e_M and e_N are the selected signals in the equilibrium of the M type sender and the N type sender, $c_M(e_M)=e_M$ and $c_N(e_N)=\gamma e_N$ are the cost of signaling of each type of

 ISBN: 978-1-61208-154-0

sender (where $0<\gamma<1$), and $w(e_M)$ and $w(e_N)$ are the fee schedules, and the separating signaling equilibrium exists and satisfies the following conditions.

$$w(e_M) = 0 \qquad w(e_N) = v \tag{1}$$

$$w(e_M) - e_M \geq w(e_N) - e_N \qquad w(e_N) - \gamma e_N \geq w(e_M) - \gamma e_M \tag{2}$$

The fee schedule $w(e)$ that satisfies the following satisfies conditions 1) and 2).

$$w(e) = \begin{cases} v & \text{for } e \geq e^* \\ 0 & \text{for } e < e^* \end{cases} \tag{3}$$

Finally, the optimal level of signal e^* must satisfy the following simple condition.

$$v \leq e^* \leq \frac{v}{\gamma} \tag{4}$$

For the above conditions, in the separating equilibrium, the receiver believes that the sender is M type (or N type) with probability one when he/she observes the signal e_M (or e_N). Therefore, the values of $w(eM)$ and $w(e_N)$ are 0 and v in accordance with the assumption of zero profit. Furthermore, the expected payoff of the M type sender is $v+L-e_M^*$ and for the N type sender it is $2v-\gamma e_N^*$. It is clear that $e_M^*=0$ is the best choice of the M type sender. For the N type sender, $e=0$ is the best choice and the payoff is only equal to v if he/she selects e satisfying $e\neq e_N^*$ and the receiver has belief $\mu_e=1$ for all e other than e^*. Therefore, the N type sender does not have an incentive to leave the separating equilibrium when the following condition is satisfied: $2v-\gamma e_N^* \geq v$, that is, $e_N^* \leq v/\gamma$.

If the M type sender wants to leave the separating equilibrium, selecting e_N^* as the signal is the best choice. However, the receiver's belief is $\mu_{eN*}=0$ in this situation so that the receiver offers only the value v as the fee for the trust-based transaction and the M type sender obtains the payoff $2v+L-e_N^*$. Therefore, the M type sender does not have an incentive to leave the equilibrium when the following condition is satisfied: $2v+L-e_N^* \leq v+L$, that is, $e_N^* \geq v$.

The meaning of condition (4) is clear. To ensure the existence of the separating equilibrium in which the M type sender does not send any signal and the N type sender sends a positive signal, e_N^* has to be sufficiently high so that the M type sender cannot pretend to be the N type sender and coincidently cannot to be so high that the N type sender cannot afford the signaling cost. In the separating equilibrium, the utility-maximizing N type sender selects $e_N^*=v$ as the best choice.

What one has to focus on is that the signaling cost structures of the two types of senders have to be distinct. However, one has to consider that not every form of signal ensures a distinct cost structure for the two types of senders.

B. Pooling Equilibrium

Proposition 2. The pooling equilibrium in which the two types of senders select the same trustworthy level as a signal is not stable if the signaling cost structure is distinct.

If the sender cannot send any signal, the receiver believes that the probability that the sender is the M type sender is prior probability π_M. The receiver participates in the transaction and suggests the fee for a trust-based transaction of $\pi_M<v/(2v+L)$ so that all types of senders take w as the fee. Similarly, if the two types of senders select the same signal e^* as their trustworthiness level, the receiver cannot obtain any information regarding the type of sender. In this situation, the belief of the receiver is the same as the prior probability that the sender is the M type sender. Furthermore, the signal e, other than e^*, must satisfy the following conditions.

$$w(e^*) = v - \pi_M(2v+L) \qquad w(e) = v - \mu_M(2v+L) \tag{5}$$

$$w(e^*) - e^* \geq w(e) - e \qquad w(e^*) - \gamma e^* \geq w(e) - \gamma e \tag{6}$$

Therefore, the receiver suggests the fee schedule $w^*=v-\pi_M(2v+L)$ when he/she observes e^*. The M type sender receives the payoff $u_M(e,w)=2v+L-\pi_M(2v+L)-e^*$ and the N type sender receives the payoff $u_N(e,w)=2v-\pi_M(2v+L)-\gamma e^*$. To ensure the pooling equilibrium in which the two types of senders select the same signal, the payoff from selecting e must not be higher than the payoff that resulted from selecting e^*. Therefore, the following two inequalities have to be satisfied: $(2v+L)(1-\pi_M)-e^* \geq (2v+L)(1-\mu_e)-e$ as the payoff condition of the M type sender, and $2v(1-\pi_M)-\pi_M L-\gamma e^* \geq 2v(1-\mu_e)-\mu_e L-\gamma e$ as the payoff condition of the N type sender.

The M type sender has an incentive to leave the pooling equilibrium if the belief of the receiver is $\mu_e \leq v/(2v+L)$ and the condition $(2v+L)(1-\pi_M)-e^* < v+L-e$ is satisfied. The N type sender has a similar incentive.

In the pooling equilibrium, the receiver always has the belief $\mu_e=v/(2v+L)$, so that $e=0$ is the best choice for the any type of sender if the sender leaves the pooling equilibrium. The M type and N type senders take $v+L$ and v as their payoff from the transaction. Therefore, all types of senders do not have an incentive to leave the

 ISBN: 978-1-61208-154-0

equilibrium if the following two inequalities are satisfied: $(2v+L)(1-\pi_M)-e^* \geq v+L$ and $2v(1-\pi_M)-\pi_M L-\gamma e^* \geq v$.

Therefore, the pooling equilibrium is where the two types of senders select the same signal, e^* for all e^* that satisfy $e^* \leq v-\pi_M(2v+L)$. The receiver believes that the sender is the M type with probability one when he/she observes a lower signal than e^* and expects the type of the sender in accordance with the prior probability when he/she observes a higher signal than e^*.

However, this situation is not rational because the two types of senders obtain $(1-\pi_M)(2v+L)$ and $2v-\pi_M(2v+L)$ when they do not send any signal and the equilibrium payoffs are less than the no-signal payoffs. Therefore, the pooling equilibrium is not stable.

Figure 1 illustrates how the optimal choice of a sender changes by comparing the fees and the costs of trustworthy transactions by the level of signals.

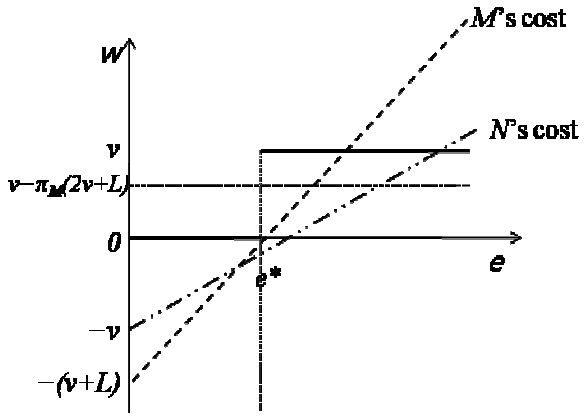


Figure 1. The fees of trustworthy transactions and costs by the level of signals.

C. The Existence Condition of the Equilibrium

Proposition 3. The effectiveness of the trustworthiness level signaling depends on the proportion of malicious type participants in the market.

When the two types of sender cannot send any signals, they obtain $(1-\pi_M)(2v+L)$ and $2v-\pi_M(2v+L)$ for each type of sender. For the N type sender, the strategy that he/she sends the equilibrium signal gives a better payoff than the strategy that he/she does not send any signal if the following inequality is satisfied: $2v-\gamma e_N^* > 2v-\pi_M(2v+L)$, that is, $\pi_M > e_N^*/(2v+L)$. This condition becomes $\pi_M > \gamma v/(2v+L)$ in accordance with the separating equilibrium condition. This means that the proportion of M type senders must be higher than a certain level so that N type senders do not achieve a better outcome when they are treated as average senders including M type senders. Additionally, the receiver wants to transact with the sender if the condition $\pi_M < v/(2v+L)$ is satisfied.

Therefore, the existence condition of equilibrium is the following inequality.

$$\gamma \frac{v}{2v+L} < \pi_M < \frac{v}{2v+L} \quad (7)$$

The most important factor in equation (7) is gamma. Gamma is the signaling cost ratio of the N type sender to the M type sender. The range of the proportion of the M type senders in the market increases as gamma increases.

For example, if the value obtained from a trust-ensured transaction, v, is equal to one for the receiver and equal to one for the N type sender and the value extorted from a deceitful transaction, $v+L$ is two (one plus one), then the equilibrium can exist when the proportion of M type senders is less than 1/3. Furthermore, if there exists a third party authority and it charges the M type sender twice the higher cost for certification of the same level of trustworthiness, the trust signaling can be effective when the proportion of M type senders is greater than 1/6.

V. AGENT-BASED SIMULATION AND RESULTS

A. The Simulation Description

This section validates the results of the equilibrium analysis. While the theoretical analysis assumes rational agents, the agent-based simulation assumes that agents have bounded rationality, which helps us understand the behaviors of real-life decision makers.

The agents that participate in the virtual market are bounded-rational receivers and senders. The signal senders are sellers that have perfect information of their own goods and their types. The signal receivers are buyers and they do not have sufficient information about the goods and the sellers' types. While there are numerous senders and receivers in the virtual market, their searching and comparing capabilities are also bounded so that they can search and compare only a few opponents. Every agent has his/her own type, selects a strategy by simple heuristics, and amends the previous strategy by evaluating the payoff resulting from the previous transaction. This is a process with behavioral elements similar to that suggested by some studies analyzing artificial intelligence.

The senders are one of two types as in the equilibrium analysis. The N type senders prefer to maintain the rules of the transaction and the M type senders prefer not to keep the rules. These senders want to increase the probability that they are selected as a partner of the receiver by sending a proper signal. The senders use the derivative follower algorithm that an agent changes their strategy based on the presented profit. This algorithm has been often used as the pricing algorithm of producers in the analysis of artificial intelligence or electronic commerce [15]. Specifically,

 ISBN: 978-1-61208-154-0

the sender changes the signal in the same direction until the current profit drops below the profit observed in the previous period and the previous profit also drops below the profit observed in the period before previous period. With these basic heuristics, the senders use additional algorithms. One of them is that although the net profit tends to increase, the agent may decrease the signal when one expects the additional profit to decrease with the signal. The signal cannot be negative.

In the simulation, the receivers are of two types. The first type of receiver prefers to transact with the sender who sends the highest signal among those agents searched by the receiver. The second type of receiver prefers to minimize the fee cost of ensuring a trustworthy transaction.

The first type of receiver thinks that the sender who has the highest signal is the N type sender, so that he/she pays the expected value that can be obtained from the transaction with the N type sender as the fee for the trustworthy transaction. If the searched signals are all of similar magnitudes, the receiver pays the expected value that can be obtained from the transaction with the average sender using the prior probability that the sender is an M type sender. These two types of receivers make decisions based on the presented signals and the transaction value *v*. Additionally, the receiver cannot punish the malicious agent.

Each run of simulation has 200 iterations. The population proportion of malicious senders in the entire population of senders varies from 0.05 to 1 for comparing the utility changes and checking simulation sensitivity. The population proportion of cost minimizing receivers in the entire receivers is set to 0.5. The gamma which means the ratio of signaling costs of two types of senders is set to 1/3 and one. The value of a good and the benefit of a sender by extortion from receiver are normalized to one. The simulation parameters are described in Table 1.

TABLE I. SIMULATION PARAMETERS

Parameters	**Value**
Iteration	200
Senders	
Number of senders	100
Proportion of malicious senders	Various
v (value of goods)	1
L (additional extortion)	1
Signaling decision algorithm	Change signal by step size
Step size	0.1*v
e_M (Signals of malicious senders)	80% of N(0,0.025*v) and 20% of N(1,0,025*v)
e_N (Signals of normal senders)	80% of N(1,0.025*v) and 20% of N(0,0,025*v)
γ, gamma	1/3, 1
Receivers	
Number of receivers	100
Proportion of cost-minimizing receivers	0.5

While the base value of malicious sender's signal is zero, 20% of malicious senders are set the initial values of signals to one in order to pretend to be normal senders as shown in Table 1. Oppositely, 20% of normal receivers are set their initial signals to zero in order to minimize their signaling costs while other receivers are set their initial signal to one. The signals are normally distributed with a mean of one or zero and a standard deviation of 0.025 to distinguish each individual signal. The signals vary stepwise with the derivative follower algorithm; the size of the step is 0.1. This value means 10 % of initial value of normal sender's signal.

B. *The Simulation Results*

Figure 2 illustrates the simulation results of the signal changes of two types of senders for various periods. The signals of the two types seem to converge before the 20th period; however, they finally diverge to around 1 and below 0.2 and become stabile.

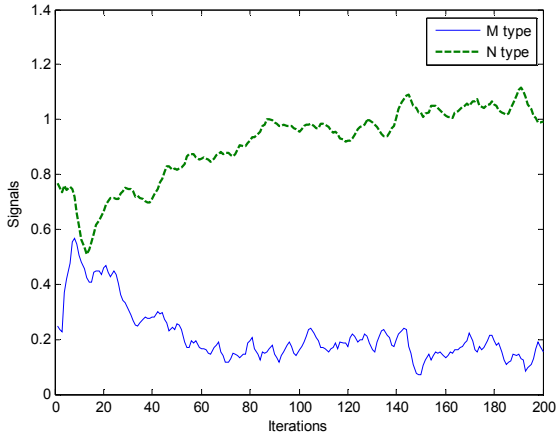


Figure 2. Signal changes of the two types of senders

From equation (7), we expect that the separating equilibrium such as in Figure 2 exists in the range from 1/9 to 1/3 for the given parameters in this simulation.

The simulation results of Table 2 indicate that if M type senders increase to 40 percent of total senders, over 80 percent of receivers take losses.

TABLE II. UTILITY CHANGE OF THE RECEIVERS

Proportion of M type senders	**0.1**	**0.2**	**0.3**	**0.4**
Average utility of receivers	0.79	0.45	–0.73	–1.30
Number of receivers having positive utility	50	36	25	16

The simulation results of Table 3 indicate that if the M type senders are less than 1/9 of total senders, the total sum of utilities in this situation is less than in the no-signaling situation.

 ISBN: 978-1-61208-154-0

TABLE III. COMPARISON OF THE SUM OF UTILITIES

Proportion of M type senders		0.05	0.1
Signaling	*M type*	2.72	3.11
	N type	2.86	2.87
	All types	2.85	2.89
No-signaling	*M type*	4.04	4.40
	N type	7.89	7.99
	All types	4.00	4.00

Finally, if the signaling costs of the two types of senders cannot be distinguished from each other, that is, gamma equals 1, the results of the simulation suggest that the M type senders increase their signal more than the N type senders, as indicated in Figure 3.

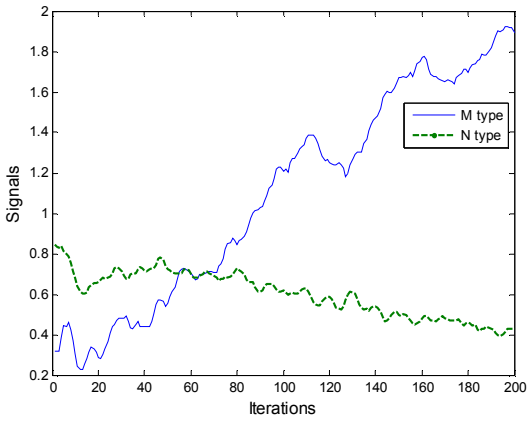


Figure 3. Signal changes of the two types of senders (when gamma equals one)

C. *The Comparison with Equilibrium Analysis*

Satisfying the criteria suggested by the results of the equilibrium analysis ensures the effectiveness of signals in distinguishing each type of participant and the stability of the separating equilibrium. First, to distinguish each type of participant on the basis of their signal, the signaling cost has to be distinct for each type to a certain extent. Second, the equilibrium that all types of participants select the same level of trust as their signal is not stable. Third, the effectiveness of distinction on the basis of the signal is affected by the revealed proportions of sender types.

The results of the simulation analysis validate the results of the equilibrium analysis and ascertain that the fundamental rules of the theoretical analysis are generally observed in the agent-based simulation in which bounded-rational agents interact with each other.

VI. CONCLUSION

This paper described the situation in which agents search, transact and manage their partners in network-based transactions based on trustworthiness signals and tried to formalize the fundamental rules of this situation using game theory, particularly the signaling game. In the situation described in this paper, the seller sends a signal of his/her trust level and the buyer decides his/her payment schedule for the presented signal. The results of the equilibrium analyses suggest the criteria of the signaling cost structures of participants and the market environment. Additionally, the results of the simulations validate the results of the equilibrium analyses.

However, this paper has the limitation as it only tried to find several fundamental rules. It only ascertained that the trust signaling cost structures of the different types of agents have to be distinguished from each other and did not suggest specific mechanisms. Therefore, future research should focus on specific mechanisms such as certification of third party authority and suggest an extended signaling game in which various types of agents interact with each other.

ACKNOWLEDGMENT

This research was supported by the KCC (Korea Communications Commission), Korea, under the CPRC (Communications Policy Research Center) support program supervised by the KCA (Korea Communications Agency). (KCA-2011-(11-941-1-005))

REFERENCES

[1] E. Chang, T. Dillon, and F. Hussain, “Trust and Reputation for Service Oriented Environment”. John Wiley and Sons, 2005.

[2] C. Shapiro and H. Varian, Information Rules: A Strategic Guide to the Network Economy. Boston: Harvard Business School Press, 1999.

[3] A. Jøsang, R. Ismail, and C. Boyd, A survey of trust and reputation systems for online service provision, Decis. Support Syst. vol. 43, 2007, pp. 618–644.

[4] C. Shahabi, F.B. Kashani, Y.S. Chen and D. McLeod, Yoda: An accurate and scalable web-based recommendation system, Proceedings of the 9th International Conference on Cooperative Information Systems, 2001, pp. 418–432.

[5] M. Head, and K. Hassanein, “Trust in e-Commerce: Evaluating the Impact of Third-Party Seals”, Quarterly Journal of Electronic Commerce, vol. 3, no. 3, 2002, pp. 307–325.

[6] G. A. Akerlof, “The market for “lemons”: Quality uncertainty and the market mechanism,” The Quarterly Journal of Economics, vol. 84, no. 3, 1970, pp. 488–500.

[7] A. M. Spence, “Job market signaling,” Quarterly Journal of Economics, vol. 87, no. 3, 1973, pp. 355–374.

[8] A. Patcha and J. Park, “A Game Theoretic Formulation for Intrusion Detection in Mobile Ad Hoc Networks, ” International Journal of Network Security, vol. 2, no. 2, 2006, pp. 131–137.

[9] F. Li and J. Wu "Hit and run: A Bayesian game between malicious and regular nodes in mobile networks", Proc. IEEE SECON, 2008, pp. 432–440.

[10] R. Axelrod, The Evolution of Cooperation, Basic Books: New York, 1984.

[11] M. A. Nowak and K. Sigmund, “Evolution of indirect reciprocity by image scoring”, Nature, vol. 393, 1998, pp. 573–577.

 ISBN: 978-1-61208-154-0

[12] B. Johnson, J. Grossklags, N. Christin and J. Chuang, “Are Security Experts Useful? Bayesian Nash Equilibria for Network Security Games with Limited Information,” Proceedings of ESORICS'2010, 2010. pp. 588–606.

[13] J. Hwang, S. Kim, H. Kim and J. Park, “An optimal trust management method to protect privacy and strengthen objectivity in utility computing services,” International Journal of Information Technology & Decision Making, vol. 10, issue 02, 2011, pp. 287-308.

[14] A. Lopez-Paredes, M. Posada, C. Hernandez, and J. Pajares, “Agent based experimental economics in signaling games in Complexity and artificial markets,” Lecture Notes in Economics and Mathematical Systems, vol. 614, S. Klaus and H. Florian, Eds. Springer Verlag berlin Heidelberg, 2008, pp. 121–129.

[15] A. Greenwald and J. Kephart, “Shopbots and Pricebots,” Sixteenth International Joint Conference on Artificial Intelligence, August, 1999, pp. 506-511.

 ISBN: 978-1-61208-154-0

A Two-Phase Security Algorithm for Hierarchical Sensor Networks

Jingjun Zhao
Department of Computer Science
North Dakota State University
Fargo, ND, USA
jingjun.zhao@my.ndsu.edu

Kendall E. Nygard
Department of Computer Science
North Dakota State University
Fargo, ND, USA
Kendall.Nygard@my.ndsu.edu

***Abstract* – We present a two-phase security system designed for hierarchical wireless sensor networks and show how it can be used to detect Denial-of-Service attacks and track harmful intruders. This type of energy-exhaustion attacks can break the connection of a sensor network and decreases its lifetime. First, we apply a Dendritic Cell Algorithm inspired by danger theory to actively detect attacked sensors that are implementing battery exhaustion attack to neighbors. Second, the system passively analyzes the tracks of the moving harmful intruders. The tracking information is used by the base station to more efficiently monitor and control the network. We adopt Markov Chain Monte Carlo methods to track the intruders and a Tabu Search technique to accelerate the searching of the final intruder tracks. The simulation results demonstrate good performance of this corrective and preventive security mechanism on detecting the malfunction sensors and tracking the intruders.**

Keywords-Wireless Sensor Networks; Dendritic Cells Algorithm; Markov Chain Monte Carlo; Tabu Search

I. INTRODUCTION

Improved wireless communication and electronics promote the development of low-power, low-cost and multifunctional sensor nodes [11]. Large numbers of such nodes can be deployed in a wireless sensor network (WSN) to sense and report data of importance. Common application areas include hospitals, homes, battle fields, and transportation systems. Security is of high importance in most WSNs. The sensors deployed typically run on batteries with limited power and computation ability. The communication channels can be unreliable and unattended operations result in vulnerability to attacks and sensor failures such as wormhole [22], denial-of-service or sinkhole attacks [23]. Traditional cryptographic security algorithms assume that all nodes are cooperative and trustworthy [12]. This assumption is not satisfied in most real-world WSN applications so that traditional security approaches may not apply.

In this paper, we focus on detecting Denial-of-Service (DoS) attacks aimed at multiplying the rate of battery exhaustion in sensors. Harmful intruders maliciously consume sensors energy by jamming a portion of a network. Local disconnection may cause the cluster heads and the base station failure of receiving correct and complete sensing data. This type of attacks is very harmful to highly critical and sensitive applications.

We adopt Artificial Immune System (AIS) approach and multiple-target tracking techniques to detect security threats in WSNs. This proactive approach of detecting malfunction sensors and harmful intruders is more effective on increasing the service lifetime of a given sensor network than passive methods that only detect dead sensors. AIS is a problem-solving methodology inspired by how biological immune systems in mammals detect pathogens and destroy them before they cause harm to the body. In real-world WSNs it is difficult to know how many invaders of different types are present. But knowing the distribution and the tracks of intruders is very helpful information for a Base Station (BS) to have for defending the network. In our work, we utilize a Multi-target tracking technique to track mobile harmful intruders. A track is a path in time-space traveled by a target [3]. The data association problem is to identify the tracks of targets from noisy observations of target positions at known points in time.

The main contribution of our work is the development of a two-phase security mechanism for WSNs with hierarchical structure by combining a Dendritic Cell Algorithm (DCA) algorithm with a multiple-target tracking algorithm and a Tabu Search technique. The two-level hierarchical sensor network that we use for our experiments has a static backbone of sparsely placed high-end sensors nodes called Cluster Heads (CH). The low-end sensor nodes belong to different clusters based on their physical position. This hierarchical structure is well-suited for large scale sensor networks and is understood to be energy preserving [10]. The DCA algorithm works on the low-end nodes and identifies malfunctioning neighbors that have been attacked by harmful intruders. The primary malfunctions include packet change, fake packet and energy-exhaustion. The ability to defend against these basic but widely existing types of attacks in a WNS makes the algorithm a good fit in practice. The multiple-target tracking algorithm implementing on Cluster Heads is used to track the mobile harmful intruders. Information about the distribution and the trajectory of harmful intruders is used by the Base Station (BS) to assess and evaluate current network defense capability. The simulation results show that the immune-inspired algorithm can effectively detect the low-end sensor nodes that have been attacked and accurately track the mobile intruders.

Figure 1 shows the two-phase security architecture. Each low-end sensor detects malfunctioning sensor nodes and reports their positions to its cluster head. When a low-end sensor receives a message it utilizes the DCA algorithm to identify whether or not the message is abnormal. Messages are reported to head node as normal or abnormal. Before implementing the multi-target tracking algorithm, a cluster head performs a K-Neighbor Query Algorithm (KNQA) to assess the credibility of received abnormal observations. This algorithm improves the accuracy of reported abnormal observation. The tracking results are ultimately sent to the base station.

 ISBN: 978-1-61208-154-0

The rest of the paper is organized as follows. Section II discusses the related works. We detail the Dendritic Cell Algorithm in Section III. In Section IV, we present the query algorithm based on the K-Nearest Neighbor (KNN) technique. In Section V, we present the MCMC algorithm for multiple-target tracking. The Tabu Search technique used to identify the tracks of the targets in a WSN is represented in Section VI. We simulate the proposed algorithms and compare the performance with the MCMCDA algorithm in Section VII. Finally, the conclusion is given in Section VIII.

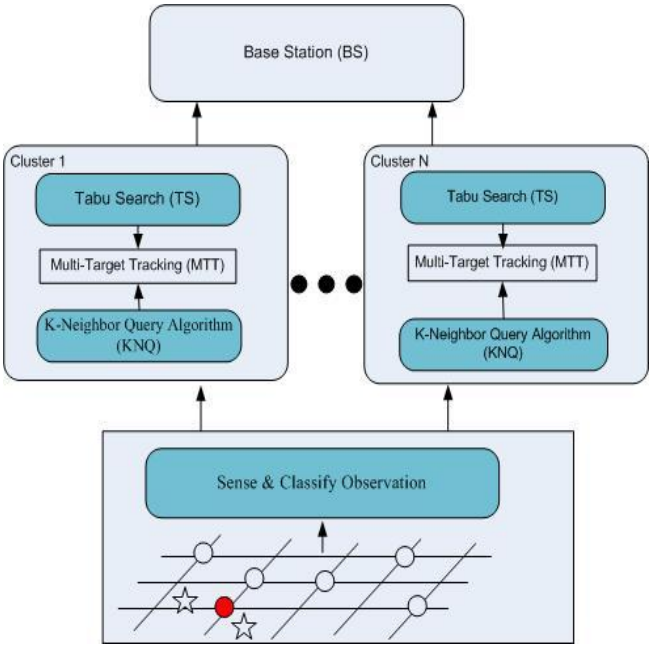


Figure 1. The two-phase security architecture (hollow circles represent normal sensors; the solid circle represents a malfunctioning sensor; and the stars represent mobile intruders)

II. RELATED WORKS

A particular class of AIS methodologies called a Negative Selection Algorithms (NSA) has been applied to anomaly detection problems [14]. Inspired by immunology, the approach uses a learning phase to construct detectors that can identify and dispatch invaders, but are not harmful to the organism itself. A fundamental issue in a NSA is maintaining distinguishing units from the host (self units) from the invaders (non-self units).[13]. Following another type of AIS, the work in [15] advanced the Danger Theory (DT) approach to intrusion detection. In Danger Theory, the central idea is that the immune system detects and responds to damage to the host, rather than upfront discrimination between self and non-self units. Dendritic cells play a central role in Danger theory. Work by Nauman and Muddassar [16] established a security system based on the behaviors of Dendritic Cells. The work reported in [17] includes detailed rules for a Dendritic Cell Algorithm (DCA) for analyzing abnormal signals. These DCAs are more flexible at detecting misbehaviors than NSAs, but do require a monitoring period to identify an intruder, resulting in inefficiency in situations with moving intruders.

Multiple-target tracking is a competent technique on tracing moving intruders in WSNs. In [3], the authors proposed a Markov Chain Monte Carlo Data Association (MCMCDA) algorithm for tracking a variable number of targets in real-time. It was established that MCMCDA is computationally efficient compared to multiple hypothesis tracker (MHT) [18], which is the prominent methodology for solving the data association problem, and outperforms MHT when there are large number of targets. MCMCDA partitions the observations into groups corresponding to candidate tracks. The collection of different partitions forms the state space for the MCMC method that searches for the most likely partitioning into intruder tracks. To make the computations of the proposal distribution easier, the authors include two additional assumptions: (1) the maximal directional speed of any target is less than $\overline{v}$; and (2) the number of consecutive missing observations of any track is less than $\bar{d}$.

Convergence rate is an important criterion on assessment of MCMC algorithm ability [20]. Although the MCMCDA makes two assumptions to decrease the size of state space, it doesn't have obvious improvement when the state space is very large. To accelerate the search of the final intruder tracks, we design two ways to improve the convergence rate of the Markov Chains used in the MCMC approach. Firstly, we further decrease the size of state space by classifying observations and predicting the moving area of an intruder. Secondly, we apply a proposed Ejection Chain algorithm to the optimal solution searching process. This approach partitions an optimization problem into relatively independent sub-optimization problems, and accelerates the optimization process. Comparing with a full MCMCDA, our algorithm needs fewer samples to reach an optimal solution.

III. THE DENDRITIC CELL APPROACH

Our previous work [5] proposed a Dendritic Cell inspired algorithm. This algorithm consists of three sub-functions: checking for dangerous messages, abnormal messages and harmful intruders. Identifying a harmful intruder is a process of changes among immature state, semi-mature state and mature state. When a sensor node receives a new message, the algorithm firstly checks a saved Harmful Intruder (HI) list. If the sender of this message is an identified harmful intruder with mature state, the message is considered to be a dangerous message. If the sender is an identified suspicious intruder with semi-mature state and exists in a Semi-Harmful Intruder (SHI) list, the algorithm implements the function of identifying harmful intruder to decide whether this sender is dangerous enough to be considered as a harmful intruder. Finally, the algorithm implements the function of checking for abnormal message. The sender is saved in the SHI list and its state changed from immature to semi-mature if the message is abnormal. This algorithm only consumes limited energy of a sensor node because of the simple calculation in each sub-function. The experiment results showed that this algorithm is capable of detecting static harmful nodes.

IV. K-NEIGHBOR QUERY ALGORITHM

In [1], the authors proposed the KNN Perimeter Tree (KPT) algorithm for dynamically processing KNN queries in location-

 ISBN: 978-1-61208-154-0

aware sensor networks. The algorithm KPT first adopts GPSR, which is a geographical routing algorithm [2], to find the Home Node (HN) that is the nearest sensor to the Query Point (QP). As by-products, the perimeter nodes around the QP are also determined when the HN is found. A circular boundary including at least K sensor nodes is determined. At each hop around the perimeter, the midpoint on the line between perimeter nodes is computed, and by plotting a line from the QP through the midpoint to the circular boundary the sub-tree boundaries are determined. In each boundary, a spanning tree is constructed, shown in Figure 2. Each node on the tree reports its position and ID to its parent node. Finally, all sensor information in the circular boundary is reported to the HN. The HN sorts these nodes by their distance to the QP to find the K nearest neighbors.

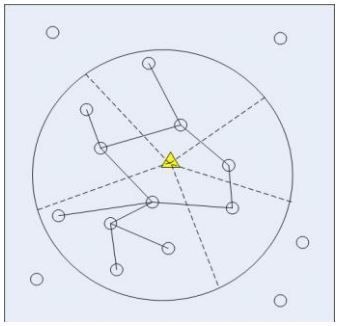


Figure 2. KNN Perimeter Tree

The perimeter boundaries can keep the spanning trees balanced and thus reduce the overall query latency. It is still possible that the K nearest neighbors are all or mostly in one spanning tree. Only querying some sensor nodes densely distributed in a small area around the QP cannot be trusted. In a given spanning tree, each node sends packages to its parent node instead of broadcasting. This reduces the number of totally transmitted message. But if a parent node in a spanning tree close to the root node has failed (e.g., because of energy exhaustion), this spanning tree will be useless for finding the K nearest neighbors.

To avoid these disadvantages and make the KNN query algorithm more efficient for detecting harmful intruders, we design the K- Neighbor Query Algorithm (KNQA) for finding K querying neighbors. Instead of constructing spanning trees in each sub-boundary, the KNQA finds K querying sensor nodes around a suspect harmful intruder, and these querying nodes tend to be closely and sparsely distributed around the suspect intruder.

Figure 3 shows an example of identifying 10 querying sensor nodes around a suspect harmful intruder. The first group querying nodes are those closest to the suspect node. We first, find the closest node to the suspect intruder, and then use a Right-Hand Rule to establish a perimeter [7], sequence of edges. The left querying nodes are searched clockwise, checking each boundary which has $\theta \geq 10^{\circ}$. In each boundary, we find the nearest node to the suspect intruder, and plot a line from the suspect intruder through the newly found node to the circular boundary. The partition of boundary continues until K nodes have been found or no boundary has $\theta \geq 10^{\circ}$.

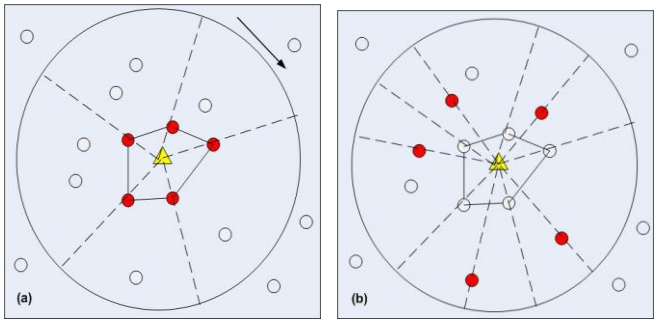


Figure 3. Querying K sensor nodes around a suspect intruder (K = 10) (a) the first 5 sensor nodes; (b) the second 5 sensor nodes

In [6], the authors described an Optimal Threshold Decision Scheme. According to the results in [6], the best policy for each node is to accept its own sensor reading if and only if at least half of its neighbors have the same reading. In this paper, we adopt this policy for deciding a harmful intruder.

V. MULTIPLE-TARGET TRACKING ALGORITHM

A. Markov Chain Monte Carlo

1) Problem formulation

In [3], the authors designed the MCMC data association (MCMCDA) algorithm for tracking an unknown number of targets that appear and disappear in the surveillance region during a surveillance period of time. The Markov chain Monte Carlo data association algorithm can initiate and terminate tracks autonomously and is robust to a high level of false alarms and missing measurements, a common problem in sensor networks [8]. During a surveillance period T, K targets appear in the surveillance region $\mathcal{R}$ for some duration $[t_a^k, t_b^k] \in [1, T]$. Each target moves in $\mathcal{R}$ at a random position at t_a^k, and moves out of $\mathcal{R}$ at t_b^k. At each time, a target disappears with probability p_z. The number of targets arriving at each time over $\mathcal{R}$ has a Poisson distribution with a parameter $\lambda_b V$ where λ_b is the birth rate of new targets per unit time, per unit volume V. Similarly, the number of false alarms has a Poisson distribution with a parameter $\lambda_f V$ where λ_f is the false alarm rate per unit time, per unit volume. The detecting probability of a noisy observation is p_d.The number of observations at time t is $n(t)$. The purpose of MCMCDA is to find out the values K and $[t_a^k, t_b^k]$ (k = 1, 2, … K).

2) MCMC Algorithm

MCMCDA adopts the Metropolis-Hastings algorithm to generate samples from a distribution π on a solution space Ω by constructing a Markov chain with state $\omega \in \Omega$ and stationary distribution $\pi(\omega)$. The acceptance probability of a proposed state ω' is defined as:

$$A(\omega, \omega') = \min\{1, \frac{\pi(\omega')q(\omega'\omega)}{\pi(\omega)q(\omega,\omega')}\} \quad (1)$$

where ω is the current state and q is the proposal distribution.

Let $y(t) = \{y^i(t): i = 1, 2, \ldots, n(t)\}$ be all observations at time t and $Y = \{ y(t): 1 \leq t \leq T \}$ be all observations during the surveillance of T. The solution space Ω is defined to be a collection of partitions of observations Y, for $\omega \in \Omega$:

(1) $\omega = \{\tau_0, \tau_1, \ldots, \tau_K\}$;
(2) $Y = \cup_{k=0}^{K} \tau_k$ and $\tau_i \cap \tau_j = \phi$ for $i \neq j$;
(3) τ_0 is a set of false alarms;
(4) $|\tau_k \cap y(t)| \leq 1$ for $k = 1, 2, \ldots, K$ and $t = 1, 2, \ldots, T$; and
(5) $|\tau_k| \geq 2$ for $k = 1, 2, \ldots, K$.

The MCMCDA defines the stationary distribution $\pi(\omega)$ as:

$$P(\omega|Y) \propto P(Y|\omega) * P(\omega) \quad (2)$$

where

$$P(\omega) = \prod_{t=1}^{T} p_z^{z(t)}(1-p_z)^{c(t)} p_d^{d(t)}(1-p_d)^{g(t)} \lambda_b^{a(t)} \lambda_f^{f(t)}$$

$$P(Y|\omega) = \prod_{\tau \in \omega \setminus \{\tau_0\}} \prod_{i=1}^{|\tau|-1} \mathcal{N}(\tau_{i+1}|\mu, \sigma)$$

In this framework, the tracking problem is toto find a state ω^* with the maximum posterior among all of the checked states.

$$\omega^* = \arg\max(P(\omega|Y)) \quad (3)$$

The Kalman filter is used to estimate the expected value μ and covariance σ. $P(Y|\omega)$ is the likelihood of observation; $z(t)$: the number of targets terminated at time t; $a(t)$: the number of new targets at time t; $d(t)$: the number of actual targets detected at time t; $e(t-1)$: the number of targets from time $t-1$; $c(t) = e(t-1) - z(t)$: the number of targets from time $t-1$ that has not terminated at time t; $g(t) = c(t) + a(t) - d(t)$: the number of undetected targets; $f(t) = n(t) - d(t)$: the number of false alarms.

B. Reduction of State Space Size

In [3] the authors make assumptions that any target has a maximal directional speed $\bar{v}$, and that the number of consecutive missing observations of any track is less than $\bar{d}$. To further accelerate the convergence rate of the Markov Chain, we distinguish abnormal from normal observations and identify the moving scope of an intruder at each monitoring time.

1) Distinguish abnormal observation from normal observation

When a sensor node receives a modified or fake message from an intruder it reports an abnormal observation to the cluster head. otherwise reports a normal observation. The state space is a collection of partitions of observations. Each partition has a number of tracks that consist of different observations. Each track can only include normal observation or abnormal observation, which reduces the size of the state space. Figure 4 shows an example of partition with classified observations.

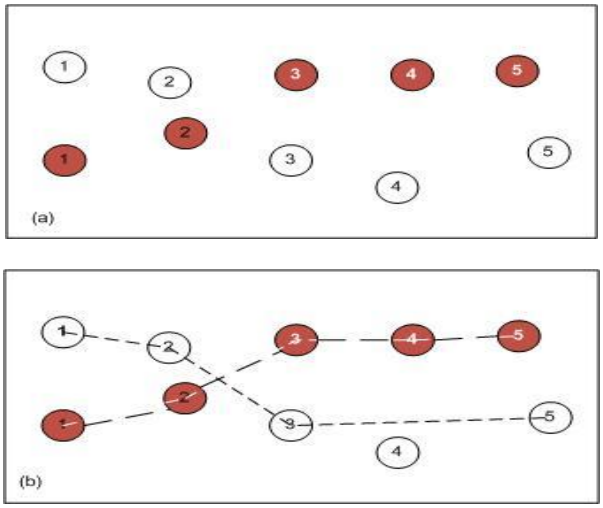


Figure 4. (a) an example of observation Y (solid circles represent abnormal observations and hollow circles represent normal observations. The numbers represent observation times); (b) an example of a partition ω of Y

2) Forecasting intruder position

We follow the method of [4], where it is assumed that a sensor node can sense an intruder approaching or moving away. We achieve this function by computing the energy level of signals, which requires small computational power [9].

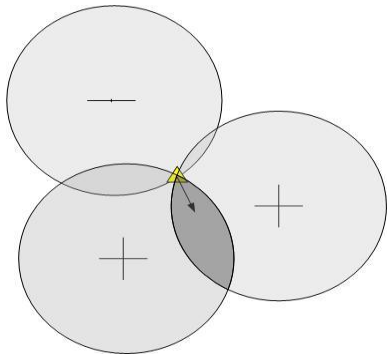


Figure 5. The future moving direction of an intruder within the shaded area

A sensor reports the movement trend of an intruder to the cluster head along with the normal or abnormal observation. The cluster predicts the movement of an intruder at a future time by collecting and analyzing received information from different sensors at a synchronized time and at the same position. Figure 5 illustrates three sensors. Two of them sense that an intruder is approaching them (+ symbol), and one senses that the intruder is moving away (- symbol). The triangle shows the intruder position. The prediction is that the future position of this intruder is in the shaded area.

VI. TABU SEARCH

We utilize a local search technique to sample from the obtained observations. The sampling process is divided into two steps. First, an initial feasible set of tracks is constructed. Second, an improved new neighboring set of tracks is found. We define a cost function for each track as below:

 ISBN: 978-1-61208-154-0

$$f(\tau_k) = \sum_{i=1}^{n-1}\left(\lambda d_{i,i+1} + \lambda^2 p_{i,i+1} + \lambda^3 t_{i,i+1}\right) \quad (k = 1 \ldots K) \tag{4}$$

We use τ_k for the kth track; n is the number of observations in track τ_k ; $d_{i,i+1}$, $t_{i,i+1}$ and $p_{i,i+1}$ are Boolean values representing if the distance of two sequential observations exceeds a threshold value; if the types of two sequential observations are identical; and if the relative position of two sequential observations is in the forecasted area (the shaded area in Figure 5).

An improved track must have cost that is no larger than the cost of the old track. The new neighboring set of tracks can include one or more than one improved tracks according to actual requirements. This new neighboring solution is used as a proposed state to calculate the acceptance probability in the MCMC algorithm.

A neighborhood structure for defining moves based on ejection chains is well-known for the traveling salesman problem [21]. We use neighborhoods defined by ejection chains to produce effective moves with reduced computational effort. This method is a local search optimization technique which tries to minimize a cost function F(x), where x represents a parameter vector, by iteratively moving from a solution x to a solution x0 in the neighborhood of x until a stopping criterion is satisfied or a predetermined number of iterations N is reached. Algorithm 1 and 2 show the ejection chain and Tabu search methodologies.

We developed an ejection chain algorithm for the Tabu Search (TS) framework to determine if a trial set of tracks from received observations during a surveillance period. We group the observations by time period, then identify trial solutions in each group of observations using a proposed ejection chain algorithm. A trial solution is a connection among observations in a group. Ultimately, we combine all these group trial solutions to get a final trial set of tracks. Figure 6 and 7 illustrate the method. This approach partitions an optimization problem into relatively independent sub-optimization problems, and accelerates the optimization process.

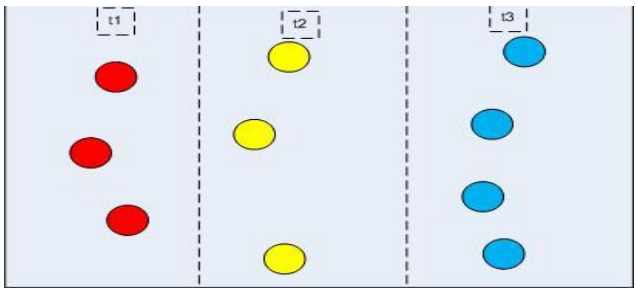
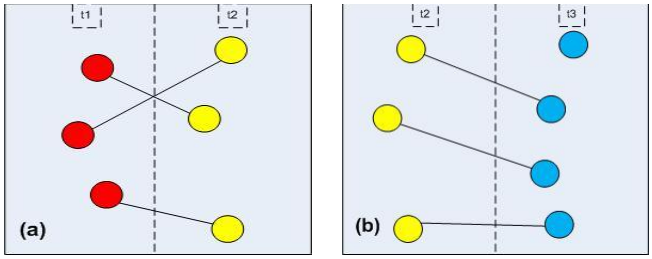


Figure 6. An example of observation Y (from time t1 to t3)



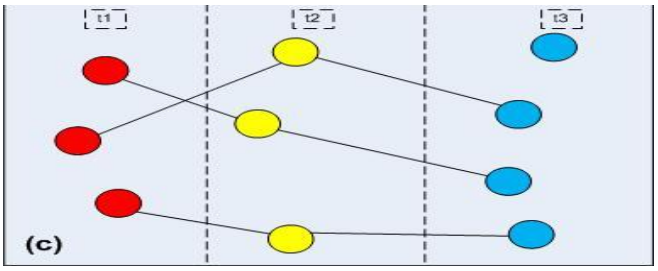


Figure 7. Illustration of searching for a trial set of tracks from observation Y (a) trial solution of group1; (b) trial solution of group2; (c) the whole trial solution

Algorithm 1 Ejection Chain Algorithm (ECA)

$n1$: Number of observation in the first column;
$n2$: Number of observation in the second column;
L: Current Ejection level;
L^*: The current best ejection level;

1: Set $L = 1$, $L^* = L$.
2: Create the first level of the ejection chain
(a) Start from the node which has the largest track cost;
(b) Try to generate a new trial solution with no larger cost;
(c) If no such trial solution, go to step 4;
(d) Update current trial solution;
(e) If no ejection occurred, go to step 4;
(f) Record the last ejection node $\boldsymbol{e^L}$.
3: Increase the chain to further levels (ref [19])
(a) Set $L = L + 1$;
(b) Start from $\boldsymbol{e^L}$, determine a new element that doesn't increase the cost;
(c) Update current trial solution and $\boldsymbol{e^L}$;
(d) If $L < Max\_LEVEL$ and L $< Min(n1, n2)$ go back to step 3. Otherwise go to step 4;
4: Get a new trial solution S'; Exit.

Algorithm 2 Tabu Search Algorithm (TSA)

Temp: flag;
TM: the maximum value of iteration

1: Generate a starting solution in S randomly, let $S^* = S$;
2: Call Ejection Chain Algorithm.
3: If improving: set Temp = 0, update the best solution with the new current solution, $S^* = S'$; otherwise Temp++;
If Temp < TM return to step2, otherwise Exit.

VII. SIMULATION RESULTS

We wrote the MCMC algorithm and Tabu Search algorithm in the C++ language with Matlab interfaces, and implemented the DCA algorithm in the Java Agent DEvelopment Framework (JADE). The results of running the DCA algorithm are saved in a file that is called by the MCMC algorithm, KNQ algorithm and Ejection Chain algorithm. The surveillance area

 ISBN: 978-1-61208-154-0

is a $\mathcal{R} = [0, 100] \times [0, 100] \in \mathbb{R}^2$ rectangular region, and 200 sensor nodes are randomly deployed in the area.

Four experiments are implemented. The first experiment is used to measure and analyze the efficiency of the DCA algorithm on identifying malfunctioning sensor nodes; the second assesses the efficiency of the KNQ algorithm implemented by cluster heads. The KNQA verifies the credibility of received abnormal observations. The last two experiments show that the local search algorithm accelerates convergence of the Markov chain.

A. *Experiment 1-DCA algorithm*

We randomly deploy 10, 20, 30, 40 and 50 intruders in the test area and perform experiments on each case. In each case we take 10 samples. Figure 8 shows the results for detecting malfunction sensor nodes. The results show the DCA algorithm can detect more than 90% of the attacked sensor nodes when fewer than 10% of the intruders are harmful. The algorithm needs an initial period of time and a cache to monitor and identify a malfunctioning node. This explains why there is a low detecting rate at the beginning of the sampling times, and decreased detecting rate when there are more harmful intruders.

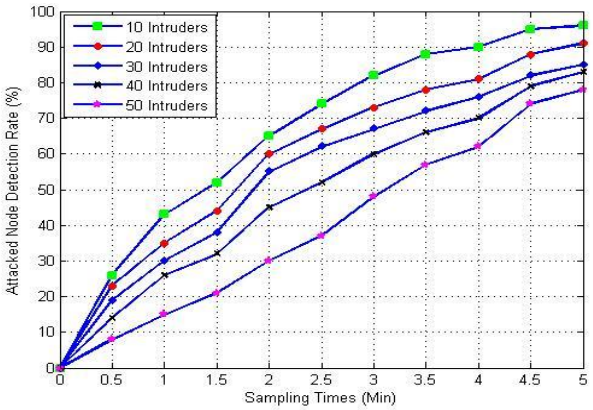


Figure 8. Sampling time vs. attacked node detection rate with different number of harmful intruders

B. *Experiment 2 – K-Neignbor Query Algorithm (KNQA)*

The accuracy probability of receiving a message from a suspect harmful intruder is defined as:

$$P_a = 1 - \frac{1}{n}\sum_{i=1}^{n} \cos\theta_i \quad (5)$$

where n is the number of neighbor nodes, and θ_i is the angle of the ith neighbor node, the suspect intruder and the current node. Only those neighbors that have $\theta_i \epsilon [0^0, 90^0]$ are considered. In this experiment, we analyze the effectiveness of the KNQ algorithm on detecting abnormal observation based on the data from the first experiment. There are five groups of data for each number of intruders in a WSN. The analysis is for the group with 20 intruders. The threshold value for the accuracy probability in (5) is 0.85. The number of querying node is 10.

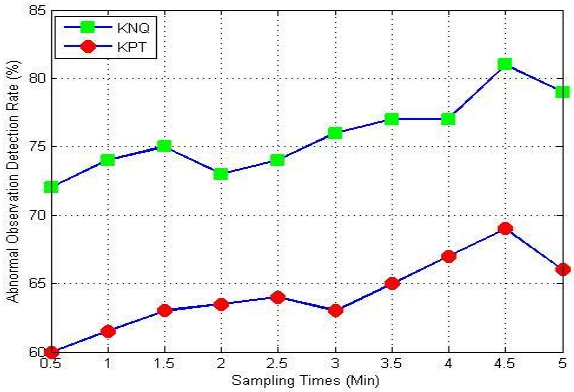


Figure 9. KNQ vs. KPT on detecting abnormal observations

Figure 9 indicates that the KNQ algorithm has higher detection rate than KPT algorithm when considering interference among neighbor nodes. At sampling time 4.5, the KNQ has an obvious higher detection rate than KPT, which is caused by some nodes densely distributed in a particular area.

C. *Experiment 3 –MCMC with Tabu Search*

In this experiment, we utilize the same simulation settings as in [3]. The surveillance area is a $\mathcal{R} = [0, 100] \times [0, 100] \in \mathbb{R}^2$ rectangular region. The number of mobile targets K varies from 10 to 50. The other parameters are: $T = 10$, $p_d = 0.8$, $\lambda_f V = 1.0$, $\lambda_b V = 1$, $p_z = 0.01$, $\bar{d} = 5$, $\bar{v} = 100$ unit lengths per unit time. The state vector is $x = [x, y, v_x, v_y]^T$ where (x, y) is a coordinate and (v_x, v_y) is a velocity vector. The Kalman filter is used to estimate the states of an target, and the models are:

$$x_t = Ax_{t-1} + Bw_k$$
$$y_t = Hx_t + v_t$$

where

$$A(\Delta) = \begin{bmatrix} 1 & 0 & \Delta & 0 \\ 0 & 1 & 0 & \Delta \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} B(\Delta) = \begin{bmatrix} \Delta^2 & 0 \\ 0 & \Delta^2 \\ \Delta & 0 \\ 0 & \Delta \end{bmatrix} H(\Delta) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}^T$$

w_k and v_k are white noises with Gaussian distributions $\mathcal{N}(0, \sigma_w^2)$ and $\mathcal{N}(0, \sigma_v^2)$ respectively, where $\sigma_w^2 = \text{diag}(100, 100)$ and $\sigma_v^2 = \text{diag}(20, 20)$. To estimate the efficiency of finding the optimal solution among a state space, we specify value 0.9 as a threshold for $P(\omega|Y)$ defined in equation (2).

1) *Number of samples vs Number of tracks*

A MCMC algorithm usually iterates some fixed number of times and the maximum posterior of a state is the optimal solution. In our case, the goal is to find the state defined in (3). We use the number of iterations required to find the state that has a larger posterior then a threshold as the criterion to evaluate convergence speed. We compare the algorithm with Markov Chain Monte Carlo Data Association (MCMCDA) by running each algorithm 10 times.

 ISBN: 978-1-61208-154-0

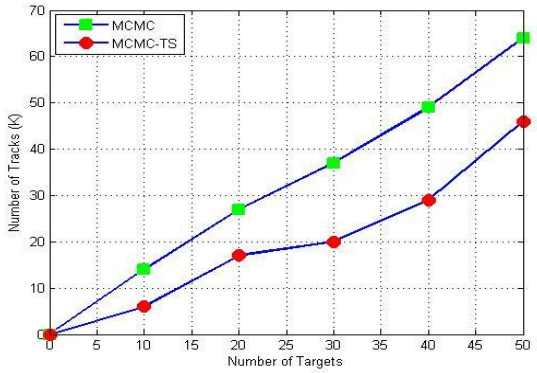


Figure 10. Number of targets vs. number of tracks

The number of tracks is significantly increased with the increase of targets. That means a longer time is needed to identify the real target tracks among the options. Figure 10 shows that the algorithm reduces the number of optional tracks.

2) Average running time vs number of tracks

The running times of the algorithms with and without state space reduction and ejection chains are shown in Figure 11. The improved MCMC procedure is significantly faster.

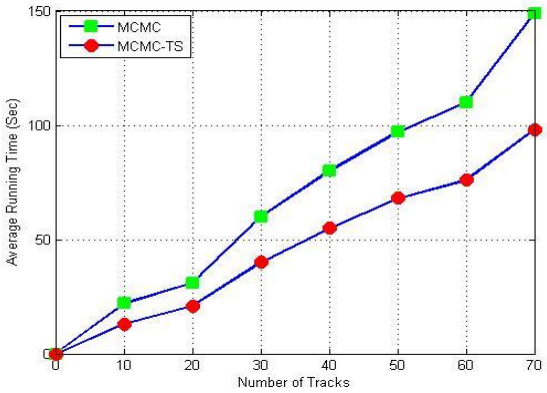


Figure 11. Average running time vs. number of tracks

VIII. CONCLUSION AND FUTURE WORK

A two-phase security mechanism that combines Dendritic Cell, MCMC and Ejection Chain algorithms was developed and tested. The DCA algorithm detects malfunctioning sensor nodes and prevents damage to a WSN by ceasing response to requests from these nodes. The MCMC procedure simultaneously tracks multiple harmful mobile intruders. Instead of randomly samplings from the reported observations, ejection chains and a Tabu Search are used together to selectively sample from the observations and accelerate the convergence rate. Our results establish that our new security mechanism promptly identifies trouble makers.

REFERENCES

[1] J.Winter and W. Lee. Kpt, "A Dynamic KNN Query Processing Algorithm for Location-aware Sensor Networks," Proc. of DMSN, 2004.

[2] B. Karp and H. T. Kung, "GPSR: Greedy Perimeter Stateless Routing for Wireless Networks," In Proceedings of the 6th Annual Int. Conference on Mobile Computing and Networking, pp. 243–254, 2000.

[3] S. Oh, S. Russell, and S. Sastry, "Markov Chain Monte Carlo Data Association for General Multiple-Target Tracking Problems," in Proc. of the IEEE International Conference on Decision and Control, Paradise Island, Bahamas, Dec. 2004.

[4] J. Aslam, Z. Butler, F. Constantin, V. Crespi, G. Cybenko, and D. Rus, "Tracking a Moving Object with Binary Sensors," Proc. ACM SenSys Conf., Nov. 2003.

[5] J. J. Zhao and K. E. Nygard, "A Dendritic Cell Inspired Security System in WSNs," FUTURE COMPUTING 2010, Int. Conference on Future Computational Technologies and Applications, Lisbon, Nov. 21, 2010.

[6] B. Krisnamachari and S. S. Iyengar, "Distributed Bayesian algorithms for fault-tolerant event region detection in wireless sensor networks," IEEE Transactions on Computers, Vol. 53, No. 3, Mar. 1, 2004.

[7] B. Karp and H. T. Kung, "GPSR: Greedy Perimeter Stateless Routing for wireless networks," in: Proceedings of the Sixth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom), Boston, MA (August 2000).

[8] P. Chen, S. Oh, M. Manzo, B. Sinopoli, C. Sharp, K. Whitehouse, G. Tolle, J. Jeong, P. Dutta, J. Hui, S. Shaert, S. Kim, J. Taneja, B. Zhu, T. Roosta, M. Howard, D. Culler, and S. Sastry, "Experiments in instrumenting wireless sensor networks for real-time surveillance," In International Conference on Robotics and Automation (video), 2006.

[9] W. Chen, J. Hou, and L. Sha, "BDynamic clustering for acoustic target tracking in wireless sensor networks," in Proc. of the 11th IEEE Int. Conf. Network Protocols, Nov. 2003, pp. 284–294.

[10] F. L. Lewis, "Wireless sensor networks," In D. J. Cook and S. K. Das, editors, Smart Environments: Technology, Protocols, and Applications. Wiley, 2004.

[11] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam and E. Cayirci, "A survey on sensor networks," IEEE Communications Magazine 40, 8 (Aug.), 102—114, 2002.

[12] J. P. Walthers, Z. Liang, W. Shi and V. Chaudhary, "Wireless Sensor Networks Security: a Survey," Report MIST-TR-2005-007, 2005.

[13] J. Kim and P. J. Bentley, "Evaluating Negative Selection in an Artificial Immune System for Network Intrusion Detection," Proc. Genetic and Evolutionary Computation Conference (GECCO), 2001.

[14] S. Forrest, S. Hofmeyr, and A. Somayaji, "Computer Immunology," Communications of the ACM, 40(10):88-96, 1997.

[15] U. Aickelin, P. Bentley, S. Cayzer, J. Kim, and J. McLeod, "Danger theory: The link between AIS and IDS," Proceedings of the Second International Conference on Artificial Immune Systems (ICARIS 2003), vol. 2787 of LNCS, Springer-Verlag; pp. 147‑155, 2003.

[16] N. Mazhar and M. Farooq, "A sense of danger: dendritic cells inspired artificial immune system for MANET security," GECCO 2008: 63-70.

[17] A. P. da Silva, M. Martins, B. Rocha, A. Loureiro, L. Ruiz, and H. C. Wong, "Decentralized intrusion detection in wireless sensor networks," in Proceedings of the 1st ACM international workshop on Quality of service & security in wireless and mobile networks (Q2SWinet '05). ACM Press, October 2005, pp. 16–23.

[18] D. B. Reid, "An Algorithm for Tracking Multiple Targets," IEEE Transaction on Automatic Control, 24(6):843–854, December 1979.

[19] C. Rego, T. James and F. Glover, " An Ejection Chain Algorithm for the Quadratic Assignment Problem," Networks, 2009.

[20] M. K. Cowles and B. P. Carlin, "Markov chain Monte Carlo convergence diagnostics: a comparative review," Journal of the American Statistical Association 91, 883–904, 1996.

[21] F. Glover, " Multilevel tabu search and embedded search neighborhoods for the traveling salesman problem," Working paper, College of Business & Administration, University of Colorado, Boulder, CO, 1991.

[22] Y. Hu, A. Perrig and D. Johnson, "Packet leashes: a defense against wormhole attacks in wireless networks," IEEE Annual Conference on Computer Communications (INFOCOM), 2003, pp. 1976–1986.

[23] A. D. Wood and J. A. Stankovic, "Denial of service in sensor networks," IEEE Computer Magazine, October 2002, pp. 54-62.

 ISBN: 978-1-61208-154-0

Cloud Computing and the Enterprise Needs for Data Freedom

Dalia Kriksciuniene
Vilnius University
Muitines str. 8, Kaunas, Lithuania
dalia.kriksciuniene@khf.vu.lt

Donatas Mazeika
Vilnius University
Muitines str. 8, Kaunas, Lithuania
donatas.mazeika@khf.stud.vu.lt

Abstract— **The article aims to investigate the problem, if the willingness and success of transferring enterprise operations to the cloud-computing infrastructure are related to the level of freedom of managing enterprise data. This problem emanates from the raising awareness of the enterprises, that submission of data for processing by the cloud computing solutions invokes risk of further transformations of data formats by vendor, and failure of keeping its compatibility to own data. In this case the enterprise could be unable to make archives, or to switch to another cloud service provider. The pilot survey explores attitudes of enterprise members and specialists of information technologies (IT) to the designed shadow service and is potential to increase customer trust for cloud computing services. The concept of shadow service architecture is suggested, which models the customized backup plan adjusted for managing data of the company as the compatibility matrix.**

Keywords-cloud computing; vendor lock-in; trust; compatibility matrix; shadow service architecture

I. INTRODUCTION

The main idea of cloud computing is to transfer on-site solutions, based on ownership and management of hardware and software products, to the IT services that can be accessed by Internet. Cloud computing solution providers have started massive marketing for enterprises encouraging them to shift from traditional IT to software on-demand. Implementation of new software solution is quite long and expensive process, while cloud computing offers service provision with lower initial costs, quick launch, and flexibility of software and hardware resources.

Cloud computing issues are widely analyzed and discussed in scholarly literature. In [1], authors define cloud computing as a large-scale distributed computing paradigm that is driven by economies of scale. Cloud computing is claimed to be a new step in internet computing that provides large perspectives, but at the same time it raises issues in the architecture, design, and exploiting of existing networks and data centers [4]. In [3] Sakr, indicates that cloud computing is the transformation of IT from a product to a service. Many authors agree that this new type of computing brings new opportunities to enterprise, such as reduced costs, improved scalability, increased customization [3], experimenting with new services, ability to improve usage by adding more capacity at peak demand, and removing unexploited capacity [4]. Together with the new promising advantages, there have emerged issues of security, data privacy, technical risks, lack of system integration, which should be solved for building trust of users.

The aim of this work is to analyze enterprise needs for ability to control its own data in cloud solutions and to investigate if services from cloud computing vendors should be modified in order to increase trust of the end-users and to avoid threat of vendor lock-in.

The following section presents cloud-computing definitions, operating principles, risks and classification of products that are used in business practice. In the third section, the Software as a service (SaaS) type of cloud solutions is explored by conducting case study, which addresses needs of the end-users and analyses how the enterprise can manage its own data on cloud server. In the fourth section the concept of shadow service is substantiated. It aims to increase trust of user by enhancing flexibility of data monitoring. The findings of the research are summarized in the conclusion section of the article.

II. BENEFITS AND RISKS OF CLOUD COMPUTING

Cloud computing refers to the various software and hardware solutions that are provided by independent vendors for access via Internet. These solutions are easily scalable and simple to start using. The payments for these services are usually made for subscription. Cloud computing offers significant computing capability and economy of scale, and recently cloud computing has become a new promising mode of business computing [5][15].

Cloud computing services can be classified to three main layers, reflecting different purposes of use: Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS) [3][8]:

- Software-as-a-Service (SaaS) layer applications are hosted by cloud computing providers and are available to customers over Internet, such as CRM, ERP, project management systems, document management systems, office suite programs etc. These solutions are targeted for business and home users.
- Platform-as-a-Service (PaaS) layer is targeted at software developers' needs, it offers both development environment and tools as a service.
- Infrastructure-as-a-Service (IaaS) layer delivers platform virtualization environment as a service [5]. Target users are system administrators, who analyse the needs for resources and ensure computing power [4].

 ISBN: 978-1-61208-154-0

IaaS and PaaS are mainly used by developers and Independent Software Vendors (ISV) [6], while SaaS services are targeted for enterprise use [4]. As company data are the critical sources of competitive advantage, the enterprises are very serious about their safety and effective use, without any obstacles emanating from the chosen cloud service.

SaaS applications can serve enterprise end-users, such as CEO, managers, administrators, for managing various business processes. The leading SaaS cloud-computing vendors are Salesforce.com, Google, IBM. The most popular SaaS solutions are the Google Apps office tools products, Salesforce.com, SugarCRM (CRM- customer relationship management systems), LotusLive (web-based collaboration tools). SaaS solutions have wide selection of functions [3]. In most cases they can be compared to analogous traditional software. They are easy to start, as they do not require specific programming or administration knowledge and could be especially suitable for small and medium enterprises that lack financial and human resources for investing to IT infrastructure: installing and maintaining hardware infrastructure and software applications [15]. By using "pay as you go" subscription model the enterprise can avoid costs of starting capital, and the running costs can be further regulated by subscribing resources and services that company needs at the time. Other benefits include scalability, reliability, security, ease of deployment, and ease of management for customers [7].

Despite of many benefits the question about getting back company's data is left open. If company subscribes some cloud services, it could be hard or impossible to make backup of all data and ensure that data could be reused of moved to other cloud vendor. One of the final conlusions of the extensive survey of 125 participants from over 100 organisations by the KPMG consulting company states that cloud computing vendors are establishing their own, partly incompatible, standards to complicate integration with other vendor's solutions [10].

Salesforce.com can serve as a case study of managing company data in cloud. In 2010, it had over 87 200 customers and was recognized as a world leading CRM provider by Gartner [12]. All the services provided by Salesforce.com are based on cloud computing. Its most known product is CRM system for sales force automation. The possibilities for the enterprise to manage its own data by using Salesforce.com CRM system can be evaluated as limited:

- It provides export possibility of all company data once a week only for subscribers of highest priced versions - Enterprise and Unlimited Editions.
- For Professional Edition this feature has to be ordered for additional fee.
- Group and Contact Manager editions do not have such feature [13].
- The provided backup is flat file format without any object relations [13].
- The additional possibility is to take enterprise data from Salesforce.com with the help of third party applications from AppExchange (application shop), yet without any guarantee that they are compatible to the current version of cloud solution for backup of company data.

The Saleforce.com case shows that one of the main disadvantages of cloud computing is the threat of locking company data's lock at one vendor and lack of integration with other systems.

Major threats of cloud computing can be summarized to the following risk categories [2]:

- Policy and organizational risks: lock-in, loss of governance, compliance challenges, loss of business reputation, and cloud service termination or failure;
- Technical risks: unavailability of service, resource exhaustion, intercepting data in transit, data transfer bottlenecks, and distributed denial of service;
- Legal risks: subpoena and e-discovery, changes of jurisdiction, data privacy, and licensing.

Policy and organizational risks can lead to difficulty of extracting data from the cloud service, and this is important reason why some companies refuse start using it [2]. It is recomended that cloud computing customers should have an alternative location for services, and the cloud provider would give proper data backup to ensure continuity even if the cloud computing provider went broke or acquired and swallowed up by a larger company [5].

Technical risks can cause short-term disorders such as services unavailability due to server or connection failure, resource exhaustion, or Denial of Service (DDoS) attacks. Major technical risk is loss of Internet connectivity [15]. Although the technical risks can be considered similar to the generic Internet-related issues, but their impact for performing business processes in cloud environment is crucial, as it ads two additional sources of risk from cloud vendor and his internet provider to the existing risks emanating at the customer side and his internet service.

Legal risks are likely to arise due to a customer data keeping in different countries. Due to different legislative systems there are risks that company's data was unintentionally disclosed, the centralization of storage and shared tenancy of physical hardware imparts more risk of unwanted data disclosure, especially in cases of cloud vendor hardware being confiscated by law enforcement agencies or through civil suits [2]. Cloud computing providers are free to offer confusing privacy policies that can be unilaterally modified at any time without notice to users [14]. It is important that users could be confident to their data security and availability upon request in order to be able to change cloud vendor for resolving any type of risk.

III. FEASIBILITY SURVEY

The pilot survey was conducted in order to reveal the point of view of customers and potential users as related to the vendor lock-in problem, to discuss issues which hinder trust, and provide insights to feasibility of new shadow service which could ensure availability of data backup from cloud upon the request from customer. This pilot survey was

 ISBN: 978-1-61208-154-0

targeted to IT-professionals, analysts, cloud providers, and users. The survey questions included:

- Expectations about cloud computing at the enterprise (Likert-type scale was applied [18]).
- Awareness of data management and level of control in cloud.
- The preference list of software solutions for most likely transfer to cloud according to trust in full control of enterprise data.

The respondents for survey were accessed virtually via business information systems related groups of LinkedIn social network site. The pilot survey consisted of 10 questions, 19 people responded to the survey. The majority of respondents (63%) indicated that they belong to IT-related industry, the other persons were from telecommunications, education, engineering, architecture, finance, banking, and insurance industries. Half of the respondents were IT professionals (53,3%), managers had a share of 26,6%, other professions were represented by one person each. The majority (75%) of respondents use SaaS cloud solutions (SalesForce.com, Google Apps, etc.). The IaaS cloud service layer was the second by popularity (18,8%). The remaining respondents did not use cloud services yet.

The question if correspondents knew about their data management abilities in cloud computing was answered differently: 47,6% respondents were fully aware of the enterprise data format applied for data download from cloud service vendor. 17,6% respondents did not know if they could safely get back their data from cloud, the remaining respondents had no practical experience in this sphere.

The perceived benefits and disadvantages of cloud computing were explored by Likert-type answer scale [18]. The highlighted benefits (strongly agree answer) were: increased mobility (56,30%), quick deployment (43,80%) and software flexibility (43,80%). Reduced costs were indicated as important benefit as well (agree answer – 43,80%). Majority of participants agreed that each type of concerns (lock-in in one vendor, cloud service termination or failure, possible unavailability of service, legal and data privacy risks) were equally important. Cloud service termination or failure was marked as one of the most important concerns (agree 41,18%; strongly agree 11,76%). The result shows that the enlisted risks are not yet resolved in satisfactory way and should be addressed in future.

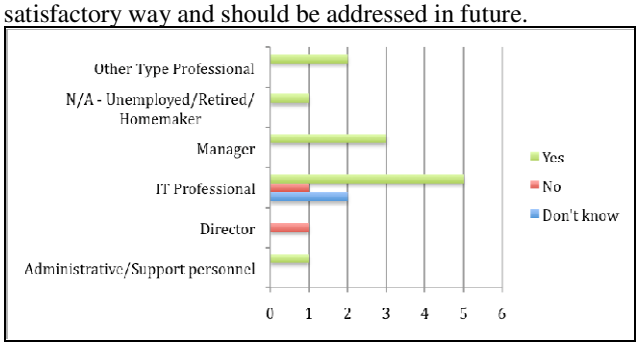


Figure 1. Question: "Does the full control of company data would increase the trust line of cloud computing?" (Distribution of answers by job roles)

The answers to question "Does the full control of company data would increase the trust line of cloud computing?" and their distribution by job roles are presented in Fig.1. The positive answers were given by 76 % of participants, who agreed that this aspect of cloud computing is very important. Only two respondents (IT professionals) answered that they don't know if the full control data management would reduce the risks. Three respondents (two IT professionals and director) expressed doubt of necessity of full data control.

The remaining two questions were to determine which of IT solutions of the enterprise the participants would transfer without hesitation to the infrastructure of cloud computing, and what types of solutions could be transferred only in case of ability to fully control company's data.

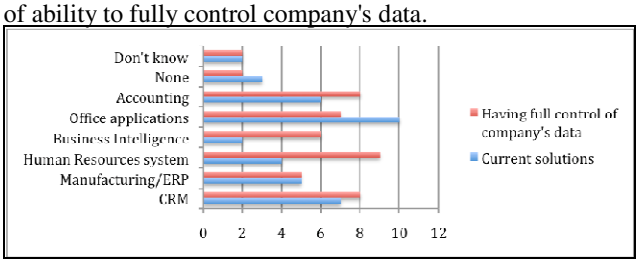


Figure 2. Question: "Which local IT solutions would you transfer to cloud?"

From Fig. 2, it is evident, that full control of company's data participants would encourage transfer of more sophisticated and customized enterprise-based IT solutions to cloud, such as accounting software, business intelligence, human resources system, CRM. Office applications were most easily considered for use in current cloud solutions.

The comments of the participants given to the question "What improvements are needed (if needed) to transform cloud computing to inevitable future model of IT?" revealed their confidence that cloud computing is the future model of IS. Security, safety, control were mentioned as absolute necessity. Less marketing, more transparent services, reviews and free market were recommended as well.

The survey confirmed that the users and potential customers are aware of benefits of cloud computing solutions for enterprise, but are strongly concerned about risks. The risks of lock-in in one vendor, cloud service termination or failure, possible unavailability of service, legal and data privacy issues were of equal importance to most participants. Software mobility, quick deployment and flexibility were the main drivers which could encourage companies for IT transfer to the cloud. The ability to control data could increase trust to cloud services and subsequently lead to transfer to cloud not only standard, but the important customized IT solutions as well. The idea for enhancement of cloud service is proposed in the following chapter as the shadow service model.

IV. SHADOW SERVICE ARCHITECTURE MODEL

In order to avoid vendor lock-in, cloud service termination or failure, the shadow service idea is suggested. New service is aimed to shadow the compatibility of interests between customer and cloud vendor and always

 ISBN: 978-1-61208-154-0

monitor enterprise data for "ready" status, which means that enterprise can access and backup its data from cloud anytime and in various formats.

In Fig. 4, the Architectural Framework for Cloud Computing is presented. It enhances basic scheme of cloud service, presented in [16]. The component of data compatibility management acts as the shadow service module, which could calibrate the interests of both customer and the cloud vendor. The shadow service has to track the amendments and changes of the cloud environment, which are performed by the vendor and to compare to the customer requirements for data formats. If the updates cloud vendor lead to further transformation of data, the rules have to be created and checked in order to inform the customer, if he can retrieve the data back from cloud in the required format for his in-site use, or possibly transfer it to the other vendor.

The shadow service module serves as a special layer, consisting of data compatibility tables and data compatibility rule engine. The data compatibility tables define the cloud service plan and compatibility matrix. The service plan is built before starting to use cloud services. The plan should include questionnaire, agreement, and other written typical documents for definition of

- Most critical data to business continuity;
- Data or documents that should be accessed without internet connection (synchronizing feature);
- Maximum retrieval time of most critical and complete data.
- Required backup formats (CSV, XML, JSON etc.).

The compatibility matrix creates the layer of interaction between the customer and cloud vendor. It serves for monitoring status of customer data within cloud and is based on preparing specific backup plans for cloud computing customers. The desired importable, processing and exportable data formats for company (or default format selection if it is not known by company) are declared here. The matrix defines the possibilities from the providers' side as well. It is used to trace changes of the customer data which could affect the compatibility of the backup data for using off-line or to transfer them to the other cloud vendor . It serves as a buffer when the connection or other technical risks arise. The shadow service idea aims for optimal data synchronization between cloud servers and company's local computers. As the full data synchronization is difficult to achieve, the priorities have to be specified for data "freshness" and readiness for backup. One of the solutions is suggested in [17] as Re:FRESHiT protocol. It specifically addresses the problems of replication management in data clouds. Re:FRESHiT organizes virtual trees based on the sites' freshness levels, and introduces a routing mechanism for reading data, while at the same time allowing users to specify their own freshness requirements [17].

Compatibility matrix-based layer serves as a set of rules for processing data and managing files according to the request of the customer and capabilities of the cloud service vendor. This feature would generally allow company to get the agreed (or default) format of data before or after using cloud services. The basic structure of compatibility matrix is shown in Fig. 3. Data compatibility management structure consists of three objects, used for describing data formats (importable data, processing data, and exportable data), and aggregated object for data compatibility management. The descriptions of objects of the compatibility matrix are filled-in when customer starts using service (either by asking customer to fill-in the questionnaire, negotiating service plan, or by offering default options).

The preferences of the customer for the importable and exportable data's formats are checked by cloud service vendor. If it is impossible for cloud vendor to import, process or export according to the requested specification, the corresponding rules within compatibility matrix are defined and further used for all company's data backup and for -local data synchronization.

Application of the shadow service and monitoring the compatibility matrix for data management could resolve problems with temporary connection breaks, because all critical data could be backuped according to the defined priorities and available for access from a local computer.

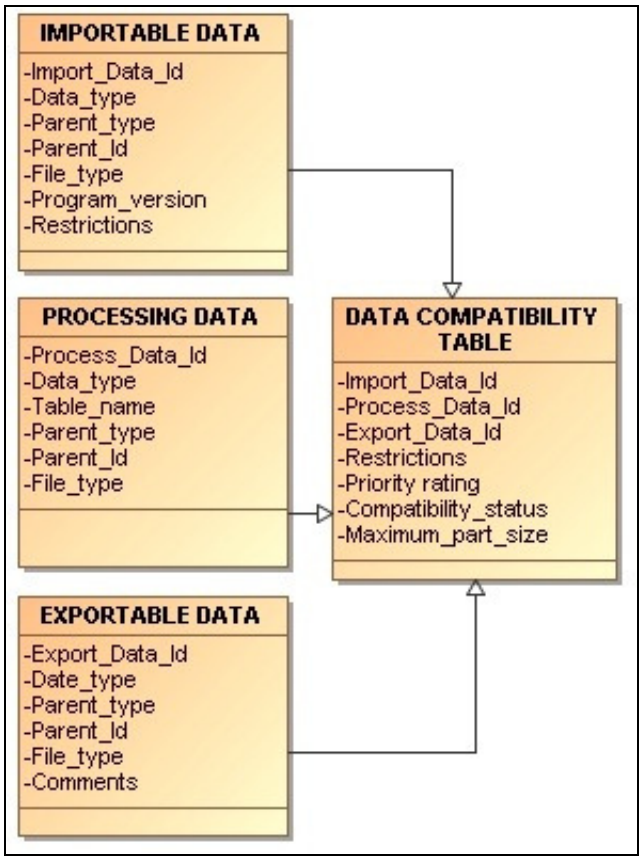


Figure 3. Compatibility matrix structure

The main idea of the compatibility matrix is to maintain importable, processing and exportable data formats, types, relations, backups, priorities by the agreed requirements. Inevitably, the cloud service provider can develop and change its environment by changing data structure or formats, but all these changes should be updated in compatibility matrix and not to violate the existing rules.

 ISBN: 978-1-61208-154-0

Data compatibility management module consists of rule engine and compatibility matrix (Fig. 3). Compatibility matrix is filled-in jointly by the customer the vendor. Some default or standard options can be applied, as well as the customized articles of mutual agreement can be introduced here. The vendor has to check the customer information and requirements about data management, and generate rules for maintaining them. In case some rules become impossible to fulfill due to changes in the cloud environment, the compatibility matrix can show the changes and require new steps from customer and the cloud vendor for altering conditions of the service agreement.

Data compatibility matrix stores information about all types of data:

Importable data: defines, what type of data and files are imported to cloud service:

- The importable data type (Date_type) could be file type or information of data array, which will further be organized to data tables.
- Parent_type and Parent_id: defines how importable data is related with another customer data, for example companies and contacts are related via company key attribute.
- File_type defines the import data source (XLS, CSV etc) or the allowed data format for upload.
- Restrictions define maximum file sizes, and requirements for files information.
- Program_version indicated the software, which was applied for creating original file.

Processing data: combines metadata information about the status of processing customer files in cloud service.

The Process_data_id and Data_type withing cloud environment are defined, changes in organizing data tables and the relations with other data (*Parent_type* and *Parent_id*), file types are set and checked by the vendor for defining rules of data compatibility.

Exportable data: area is calibrated among customer and vendor. The backup formats, relations and downloadable file types are defined.

Data compatibility table aggregated the information about importable, processing and exportable data interdependencies. Any existing restrictions, priority rates for synchronisation, statuses, maximum size and other information are filled considering mutual agreement.

The rule engine in created for managing and implementation of the compatibility matrix. If the agreement contains any particular requirement which is impossible to fulfil, it alters its status. In case some customer requirements for importable/exportable data were initialy confirmed, but later became incompatible due to changes in cloud environment, the status should change to „impossible“ or „partial“. In this case it could inform the customer about the increased risk and lead to further steps from his side or envoke obligations of the vendor to meet the requirements by customizing service.

The shadow service layer has to constantly reveal status of compatibility of interests between customer and vendor, technical status of data and information of any vulnerabilities which could hinder interests fo preserving and managing enterprise data. Transparency of techical compatibility increases customer trust and reduces risk of lock-in as well.

This shadow service enables companies to forecast further IT related strategies in order to fully control strategic data, and to reduce dependence on specific cloud service, because they know the format and integration level of enterprise data available from cloud vendor.

The suggested model brings additional costs for cloud computing vendors due to support of enhanced functionality aimed to constant track of data conversion for fulfilling requests of the customers. Their compensation would be increase in trust from clients, and building potential for solving most urgent disadvantages of cloud computing.

V. CONCLUSIONS AND FUTURE WORKS

Cloud computing vendors provide their customers various IT solutions that are easy to start using, flexible and mobile. Nevertheless the issues of concern dealing with security, data privacy, technical risks related to getting company data back from cloud in the appropriate format should be resolved for successful usage of cloud solutions.

The feasibility survey was conducted for exploring attitudes of the users and potential customers. It showed that main obstacles which hinder usage of service are related to possible cloud service termination or failure and vendor lock-in.

New shadow service architecture was suggested, which is based on creating the technical solution for monitoring agreement between cloud service and the customer vendor for defining priorities, rules and formats for managing data. The data compatibility management model, consisting of compatibility matrix and rule engine is proposed as the framework for enabling shadow service. It allows achieving transparent technical compatibility of interests between customer and services by cloud vendor, and always monitors enterprise data for “ready” status.

The compatibility matrix defines data freshness priorities, formats for data backup. It enables tracking changes performed with the data. The rule engine component enables to inform the customer, if he can retrieve the data back from cloud in the required format and ensures possibility to use the backup data with the local system of the customer and prevent from vendor lock-in situation.

The requirement for enhancement of cloud services by monitoring the shadow service layer via compatibility matrix could increase the workload and costs. But it can increase trust of customers and encourage them to transfer to cloud the most specific and strategic solutions of the company such as accounting software, customer relationship management (CRM), and human resource management systems.

REFERENCES

[1] I. Foster, Y. Zhao, I. Raicu, and S. Lu, “Cloud computing and grid computing 360-degree compared,“ Grid Computing Environments Workshop (GCE '08), 2008.

[2] T. Betcher, “Cloud Computing: Key IT-Related Risks and Mitigation Strategies for Consideration by IT Security Practitioners,” 2010.

 ISBN: 978-1-61208-154-0

[3] M.F. Sakr, “Cloud Computing/Virtualization,” The SANS (SysAdmin, Audit, Network, Security) Institute, 2010.

[4] G. Pallis, “Cloud Computing The New Frontier of Internet Computing, “ ISSN: 1089-7801, 2010.

[5] J. Yang, “Cloud Computing Research and Security Issues,” Computational Intelligence and Software Engineering (CiSE), 2010 International Conference

[6] L. Qian, Z. Luo, Y. Du, and L. Guo, “Cloud Computing: An Overview,” 2009, pp. 626-631, DOI: 10.1007/978-3-642-10665-1_63.

[7] H. Erdogmus, “Cloud Computing: Does Nirvana Hide behind the Nebula?, “ 2009, DOI: 10.1109/MS.2009.31

[8] M. Ahrens, “Cloud Computing and the Impact on Enterprise IT,” 2010, DOI: 10.1007/978-3-642-15877-3_16

[9] Salesforce, “Benefits of SaaS, “, 2010. <http://www.salesforce.com/saas/benefits-of-saas/> [last access 22/07/2011]

[10] KPMG, “KPMG is a global network of professional firms providing Audit, Tax and Advisory services,“ 2010, <http://www.kpmg.com/Global/en/WhoWeAre/Pages/default.aspx> [last access 22/07/2011]

[11] J. Schofield, “Freedom to move data is vital when it's in the clouds,“ 2009, http://www.guardian.co.uk/technology/ 2009/jun/17/cloud-computing- jack-schofield> [last access 22/07/2011]

[12] Salesforce, “Salesforce.com Positioned as a Leader in the Magic Quadrant for Sales Force Automation,“ 2010, https://www.salesforce.com/company/news-press/press-releases/2010/08/100825.jsp [last access 22/07/2011]

[13] Salesforce, “Salesforce Pricing & Editions - Sales Cloud,” 2010, http://www.salesforce.com/crm/editions-pricing.jsp, [last access 22/07/2011]

[14] R. Sprague, “Cloud Privacy: Normative Standards for Information Privacy Management within Cloud Computing. University of Wyoming College of Business,” 2009.

[15] K.R. Choo,”Cloud computing: Challenges and Future Directions” in Trends&Issues in Crime and Criminal Justica Vol.400, October 2010. Australian Institute of Criminology, 2010.

[16] R.Clarke, ”User Requirements for Cloud Computing Architecture,” Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing CCGRID '10, IEEE Computer Society, 2010, doi>10.1109/CCGRID.2010.20

[17] T. Cristiana, H. Schuldt, Y. Breitbart, and H. Schek, „Flexible Data Access in a Cloud based on Freshness Requirements," Proceedings of the 2010 IEEE 3rd International Conference on Cloud Computing CLOUD '10, IEEE Computer Society, 2010, doi>10.1109/CLOUD.2010.75

[18] R. Likert, "A Technique for the Measurement of Attitudes" in Archives of Psychology Vol 14, 1932: p.p. 1–55.

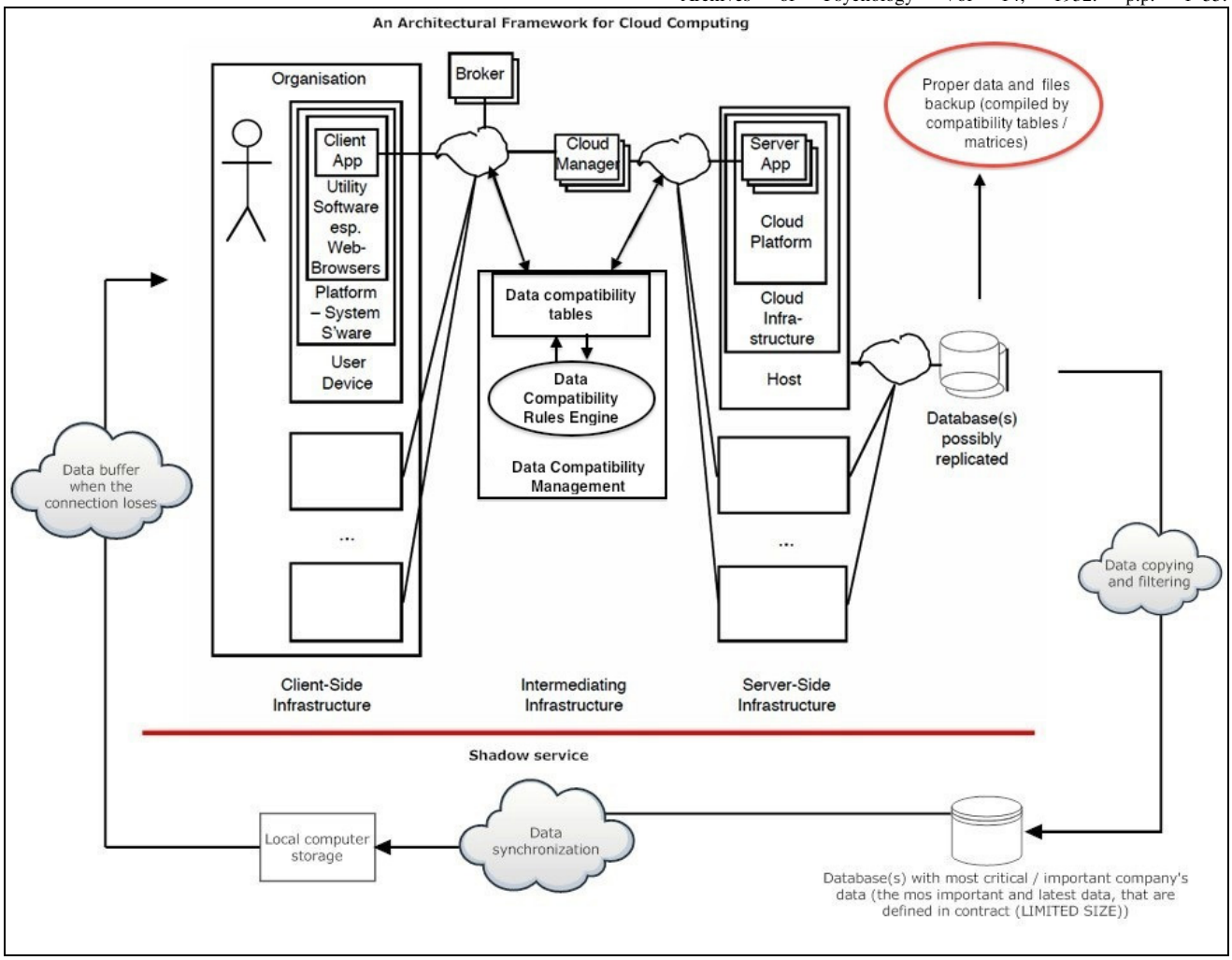


Figure 4. An Architectural Framework for Cloud computing service with compatibility layer for shadow service

A Computational Intelligence Algorithm for Simulation-driven Optimization Problems

Yoel Tenne
Faculty of Engineering,
Kyoto University, Japan
yoel.tenne@ky3.ecs.kyoto-u.ac.jp

Kazuhiro Izui
Faculty of Engineering,
Kyoto University, Japan
izui@prec.kyoto-u.ac.jp

Shinji Nishiwaki
Faculty of Engineering,
Kyoto University, Japan,
shinji@prec.kyoto-u.ac.jp

***Abstract*—Modern engineering design optimization often replaces laboratory experiments with computer simulations, resulting in what is commonly termed as *expensive black-box optimization problems*. In such problems, there will often exist candidate solutions which 'crash' the simulation and can thus lead to search stagnation and to a poor final result. Existing approaches to handle such solutions include discarding them altogether or assigning them a penalized fitness, but such approaches have significant demerits. Accordingly, this paper explores the fusion of a classifier into the optimization search to predict which solutions are expected to crash the simulation, and uses a modified objective function to bias the search towards valid ones, namely, which are expected *not* to crash the simulator. The further improve its performance, the proposed algorithm also continuously selects during the search an optimal type of classifier out of a family of candidates. To ensure the progress of the optimization search, it also employs a trust-region approach. Performance analysis using a representative real-world shape design optimization test case shows the efficacy of the proposed algorithm.**

***Keywords*-expensive optimization problems, computational intelligence**

I. INTRODUCTION

In the modern design process, engineers often replace laboratory experiments with *computer simulations*. This transforms the design process into an optimization problem having three distinct characteristics [20]:

- The simulation acts as the objective function, assigning candidate designs their corresponding objective values. However, the simulation is often a legacy or a commercial code available only as an executable, and so there is no analytic expression for this "objective function". Accordingly, it is referred to as a *black-box function*, and requires using gradient-free optimizers.
- Each simulation run is *computationally expensive*, that is, having a lengthy execution time, and this severely restricts the number of evaluations allowed during the optimization search.

and

- Often, both the underlying physics being modelled, and the numerical simulation, may result in a complicated, multimodal objective landscape, which makes it difficult to locate an optimum.

A promising optimization strategy for such expensive black-box problems is to couple a computational intelligence (CI) optimizer, which is gradient-free and handles complicated landscapes well, with *models*, namely, mathematical approximations of the true expensive objective function, but which are significantly cheaper to evaluate. During the optimization, the model replaces the expensive function (simulation), and economically provides the CI optimizer with approximate objective values.

While this approach works well, in practise another difficulty arises, as often some candidate solutions will cause the simulation to fail. We refer to such vectors as *simulator-infeasible* (SI), while those for which the simulation completes successfully and provides the objective value are *simulator-feasible* (SF). SI vectors have two main implications for the optimization problem: a) as they do not have an objective value assigned to them, since the simulation has crashed, the objective function becomes discontinuous, which increases the optimization difficulty, and b) they can consume a large portion of the limited number of calls to the expensive simulation without providing new information to the optimizer, thus potentially leading to search stagnation and a poor final result. Numerous papers, for example [1, 15, 16], have mentioned the difficulties SI vectors introduce, and so it is important effectively to handle them. Common strategies include discarding them altogether, or incorporating them into the model with a penalized fitness. However, both of these strategies have significant demerits, for example, they discard information which can be beneficial to the search, or they result in a model with a severely deformed landscape.

Accordingly, to effectively handle such SI vectors in model-assisted optimization, this paper explores the fusion of a classifier into the optimization search, and offers two main contributions. First, a classifier is used to predicts if a new vector is SI or not, and the proposed algorithm then combines this prediction with the model's prediction to bias the search to vectors predicted to be SF via a dedicated modified objective function. Second, to further enhance performance, the proposed algorithm continuously selects during the search an optimal classifier type from a family of candidates. To ensure convergence to an optimum of the true expensive function, the proposed algorithm also employs a trust-region (TR) approach. Performance analysis using an engineering

 ISBN: 978-1-61208-154-0

design application of airfoil shape optimization shows the efficacy of the proposed algorithm. The remainder of this paper is as follows: Section II describes existing approaches and open challenges in handling SI vectors, Section III describes the proposed algorithm and Section IV gives a detailed performance analysis. Lastly, Section V summarizes this paper.

II. Existing Approaches and Challenges

As mentioned in Section I, SI vectors are a common in simulation-driven optimization problems, and numerous studies mention their existence and the difficulties they introduce, for example [1, 5, 12, 15, 16].

As such vectors are both common in real-world applications, and pose the risk of hampering the optimization search, several approaches have been proposed to handle them. In reference [17], the authors used an evolutionary algorithm (EA) as the optimizer and proposed using a classifier to screen vectors before evaluating them. Those predicted to be SI were assigned a 'death penalty', that is, a fictitious and highly penalized objective value, to quickly eliminate them from the population. The study did not consider models, and the EA evaluated the expensive function directly. In a related approach described in reference [5], severely penalized SI] vectors and incorporated them into the model in order to bias the search away from them. Alternatively, in reference [1] the authors proposed to completely discard SI vectors from the training set of the model.

However, within the domain of model-assisted optimization such approaches suffer from several shortcomings: a) assigning SI vectors a fictitious penalized objective value and then incorporating them into the training set can result in a model with a severely deformed landscape, while b) excluding SI vectors altogether discards information which may be beneficial to the optimization search. As an example, Figure 1 compares two Kriging models of the Rosenbrock function: (a) shows a model trained using 30 vectors which were all SF, while (b) shows the resultant model when the sample was augmented with 20 SI vectors which were assigned a penalized fitness, taken as the worst objective value from the baseline sample. The resultant model has a landscape which is severely deformed and contains many false optima, making it difficult to locate an optimum of the true objective function.

Such issues have motivated alternative approaches to handle SI vectors in model-assisted optimization. For example, in reference [19] the authors proposed a dual model approach, where one model interpolated the objective function and the other interpolated the penalty value between SF and SI vectors. Other studies have explored the use of classifiers for constrained non-linear programming (not focusing on SI vectors), for example reference [6]. Further exploring the use of classifiers, reference [21] presented preliminary results with a classifier-assisted algorithm for handling SI vectors.

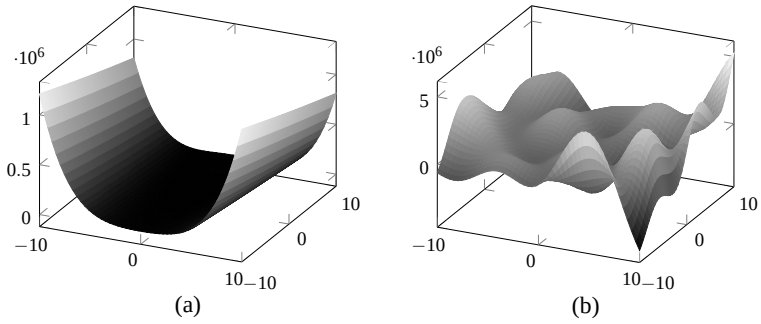


Fig. 1. Kriging models of the Rosenbrock function trained using: (a) a baseline sample of 30 vectors all SF, and (b) the baseline sample augmented with 20 SI vectors which were assigned the worst objective value from the baseline sample.

III. Proposed Algorithm

To address the issues introduced by SI vectors, as discussed in Section I and Section II, this paper proposes a model-assisted CI algorithm which incorporates a classifier into the search. To further improve the search efficacy, the algorithm continuously selects during the search an optimal type of classifier, out of a prescribed family of candidates. To ensure convergence to an optimum of the true expensive function, the algorithm also employs a TR approach. The following sections describe the model, the candidate classifiers, the classifier selection stage, and lastly, the overall workflow of the algorithm.

A. Modelling

As mentioned, the proposed algorithm uses a model to approximate the true expensive objective function. The algorithm does not impose any restrictions on the model type, and in this study the well-established Kriging model was used [11]. This model takes a statistical approach to interpolation by combining two components: a 'drift' function, which is a global coarse approximation to the true expensive function, and a local correction based on the correlation between the interpolation points. Given a set of evaluated vectors, $\mathbf{x}_i \in \mathbb{R}^d$, $i = 1 \ldots n$, the Kriging model is trained such that it exactly interpolates the observed values, that is, $m(\mathbf{x}_i) = f(\mathbf{x}_i)$, where $m(\mathbf{x})$ and $f(\mathbf{x})$ are the model and true objective function, respectively. Using a constant drift function, as in reference [11], gives the Kriging model

$$m(\mathbf{x}) = \beta + \kappa(\mathbf{x}), \tag{1}$$

with the drift function β and local correction $\kappa(\mathbf{x})$. The latter is defined by a stationary Gaussian process with mean zero and covariance

$$Cov[\kappa(\mathbf{x})\kappa(\mathbf{y})] = \sigma^2 \mathscr{R}(\theta, \mathbf{x}, \mathbf{y}), \tag{2}$$

where $\mathscr{R}$ is a user-prescribed correlation function. A common choice for the correlation is the Gaussian function [11], which is defined as

$$\mathscr{R}(\theta, \mathbf{x}, \mathbf{y}) = \Pi_{i=1}^{d} \exp\left(-\theta (x_i - y_i)^2\right), \tag{3}$$

and combining it with the constant drift function transforms the model from Equation (1) into the following form

$$m(\mathbf{x}) = \hat{\beta} + \mathbf{r}(\mathbf{x})^{\mathrm{T}}\mathbf{R}^{-1}(\mathbf{f} - \mathbf{1}\hat{\beta}). \quad (4)$$

Here, $\hat{\beta}$ is the estimated drift coefficient, $\mathbf{R}$ is the symmetric matrix of correlations between all interpolation vectors, $\mathbf{f}$ is the vector of objective values, and $\mathbf{1}$ is a vector with all elements equal to 1. $\mathbf{r}^{\mathrm{T}}$ is the correlation vector between a new vector $\mathbf{x}$ and the sample vectors, namely,

$$\mathbf{r}^{\mathrm{T}} = [\mathscr{R}(\theta, \mathbf{x}, \mathbf{x}_1), \ldots, \mathscr{R}(\theta, \mathbf{x}, \mathbf{x}_n)]. \quad (5)$$

The estimated drift coefficient $\hat{\beta}$ and variance $\hat{\sigma}^2$ are obtained from

$$\hat{\beta} = \left(\mathbf{1}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{1}\right)^{-1}\mathbf{1}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{f}, \quad (6a)$$

$$\hat{\sigma}^2 = \frac{1}{n}\left[(\mathbf{f} - \mathbf{1}\hat{\beta})^{\mathrm{T}}\mathbf{R}^{-1}(\mathbf{f} - \mathbf{1}\hat{\beta})\right]. \quad (6b)$$

Fully defining the model requires the correlation parameter θ, which is commonly taken as the maximizer of the model likelihood. In practise, maximizing the model likelihood is performed by minimizing the negative logarithm of the likelihood, which is given by

$$\mathscr{L} = -\left(n\log(\hat{\sigma}^2) + \log(|\mathbf{R}|)\right). \quad (7)$$

B. Candidate Classifiers

Given a set of input vectors with corresponding *labels*, that is, indices assigning them to a group, the goal of a classifier is to map a new input vector into one of the existing groups based on some similarity between the new and existing vectors [23]. Mathematically, given a set of vectors $\mathbf{x}_i \in \mathbb{R}^d$, $i = 1 \ldots n$, with corresponding class labels, for example, $F(\mathbf{x}_i) \in \mathbb{I} = \{-1, 1\}$, a classifier $c(\mathbf{x})$ is the mapping

$$c(\mathbf{x}) : \mathbb{R}^d \to \mathbb{I}. \quad (8)$$

The proposed algorithm puts no restriction on the number or the type of candidates classifiers, and in this study we used three well-established classifier variants [23]:

- *nearest neighbour* (NN): The classifier assigns the new vector the class of the nearest training vector, designated $\mathbf{x}_{\mathrm{NN}}$, where the latter is determined by a distance measure such as the l_2 norm, namely,

$$\begin{aligned} c(\mathbf{x}) &= F(\mathbf{x}_{\mathrm{NN}}) : \\ d(\mathbf{x}, \mathbf{x}_{\mathrm{NN}}) &= \min_{i=1\ldots n} d(\mathbf{x}, \mathbf{x}_i). \end{aligned} \quad (9)$$

  An extension of the approach, termed k nearest neighbours (k-NN) assigns the class most frequent among the k nearest neighbours. In this study the classifier used $k = 3$.
- *linear discriminant analysis* (LDA): In a two-class problem, where the class labels are $F(\mathbf{x}_i) \in \mathbb{I} = \{-1, +1\}$, the classifier attempts to model the conditional probability density functions of a vector belonging to each class, where the latter functions are assumed to be normally distributed. The classifier considers the separation between classes as the ratio of: a) the variance between classes, and b) the variance within the classes , and obtains a vector $\mathbf{w}$ which maximizes this ratio. The vector $\mathbf{w}$ is orthogonal to the hyperplane separating the two classes. A new vector, $\mathbf{x}$, is classified based on its projection with respect to the separating hyperplane, that is,

$$c(\mathbf{x}) = \mathrm{sign}\,(\mathbf{w} \cdot \mathbf{x}). \quad (10)$$

- support vector machine (SVM): The classifier projects the data into a high-dimensional space where it can be more easily separated into disjoint classes. In a two-class problem, with $F(\mathbf{x}_i) \in \mathbb{I} = \{-1, +1\}$, an SVM classifier tries to find the best classification function for the training data. For a linearly separable training set, a linear classification function is the separating hyperplane passing through the middle of the two classes. Once this hyperplane has been fixed, new vectors are classified based on their relative position to this hyperplane, that is, whether they are "above" or "below" it. Since there are many possible separating hyperplanes, an SVM classifier adds the condition that the hyperplane should maximize the distance between the hyperplane and the nearest vectors to it from each class. This is accomplished by maximizing the Lagrangian

$$L_P = \frac{1}{2}\|\mathbf{w}\| - \sum_{i=1}^{n}\alpha_i F(\mathbf{x}_i)(\mathbf{w} \cdot \mathbf{x}_i + b) + \sum_{i=1}^{n}\alpha_i, \quad (11)$$

  where n is the number of samples (training vectors), $F(\mathbf{x}_i)$ is the class of the ith training vector, and $\alpha_i \geqslant 0$, $i = 1 \ldots n$, are the Lagrange multipliers, such that the derivatives of L_P with respect to α_i are zero. The vector $\mathbf{w}$ and scalar b define the hyperplane.

C. Classifier Selection

As mentioned, during the optimization search the proposed algorithm continuously selects an optimal classifier out of a family of candidates. To accomplish this in a mathematically rigorous way, the proposed algorithm leverages on statistical model-selection theory and selects a classifier using an *accuracy estimator*. Specifically, the accuracy of each candidate classifier is estimated, and the classifier chosen is the one having the best estimated accuracy. The procedure, termed cross-validation (CV), proceeds as follows. A cache, which contains vectors evaluated with the expensive function, is split into a *training set* and a *testing set*, in a 80–20 ratio. A candidate classifier is trained using the former set and its prediction is tested on the latter set, where the classification error is

$$e = \sum_{i=1}^{l}\left(\hat{c}(\mathbf{x}_i) \neq F(\mathbf{x}_i)\right), \quad (12)$$

where $\hat{c}$ is the prediction of the trained classifier, $\mathbf{x}_i$, $i = 1 \ldots l$, are the CV testing vectors, and $F(\mathbf{x}_i)$ is the true and known class of the latter vectors, where in this study $F(\mathbf{x}_i) = 1$ was used for a SF vector, and $F(\mathbf{x}_i) = -1$ for a SI one. These class labels were used as they are common in literature, for example

 ISBN: 978-1-61208-154-0

[23]. As mentioned in Section III-B, in this study, the proposed algorithm selects between three well-established classifiers, namely *k*-NN, LDA, and SVM. The algorithm selects the classifier type having the smallest prediction error, as defined in Equation (12), and then uses all the cached vectors, namely, both SF and SI, to train a new classifier with the selected type, which serves as the classifier during the TR trial step, as explained in the following section.

D. Workflow

The algorithm begins by sampling a set of vectors which will serve as the initial training sample. The vectors are generated using the Latin hypercube design (LHD) method for experiments design [14], as it provides a space-filling sample which improves the accuracy of the model. Briefly, for a sample of k vectors the range of each variable is split into k equal intervals, and one point is sampled at random in each interval. Next, a sample point is selected at random (without replacement) for each variable, and these samples are combined to give a vector. This procedure is repeated for k times to generate the complete sample. After generating the sample, the vectors are evaluated with the expensive function and are then cached.

The main optimization loop then begins, where the algorithm first trains a Kriging model using all the SF in the cache. It then uses the procedure described in Section III-C to select a classifier type, and then trains a classifier using all the cached vectors, namely, both SF and SI, as these are two vector classes.

Next, the proposed algorithm performs an optimization search, and to ensure convergence to an optimum of the true expensive function it follows the trust-region (TR) approach. Specifically, the TR is the region where the model is assumed to be accurate, and defined as

$$\mathscr{T} = \{\mathbf{x} : \|\mathbf{x} - \mathbf{x}_\mathrm{b}\|_2 \leqslant \Delta\}, \tag{13}$$

where $\mathbf{x}_\mathrm{b}$ is the best vector found so far, and is taken as the TR centre, and Δ is the TR radius. The proposed algorithm seeks the optimum of the model in the TR, and as the optimizer it uses the real-coded EA from reference [2]. Since evaluating the model is computationally cheap, the EA uses a large population and many generations to improve its search, and Table I gives the complete EA parameter settings. During this optimization trial-step the EA does not use the model predictions directly, but instead it obtains the fitness values from the following *modified objective function*, defined as

$$\bar{m}(\mathbf{x}) = \begin{cases} m(\mathbf{x}) & \text{if } c(\mathbf{x}) \text{ is SF} \\ p & \text{if } c(\mathbf{x}) \text{ is SI} \end{cases} \tag{14}$$

where $m(\mathbf{x})$ is the model prediction, and p is a penalized fitness taken to be the worst function value from the initial Latin hypercube (LH) sample. As such, the EA receives the model prediction if the classifier predicts a vector is SF, but receives the penalized fitness otherwise. In this setup, the information about SI vectors is preserved in the classifier, but they are *not* incorporated into the model with a penalized fitness and hence do not deform its landscape.

The optimum found by the EA, designated as $\mathbf{x}^\star$, is then evaluated with the true expensive function at a cost of one function evaluation, which provides $f(\mathbf{x}^\star)$. Following the classical TR framework [3], the algorithm manages the model and TR based on the outcome of the trial step, as follows:

- *A successful trial step*: The trial step located a new optimum which is better than the current best, that is, $f(\mathbf{x}^\star) < f(\mathbf{x}_\mathrm{b})$. Following the classical TR approach, the new optimum is taken as the new the TR centre, and the TR is enlarged by doubling its radius.
- *An unsuccessful trial step*: The trial step did not locate a new optimum, that is, $f(\mathbf{x}^\star) \geqslant f(\mathbf{x}_\mathrm{b})$. This can happen since either the TR is too large, or since there are not enough vectors in the TR, resulting in a model which is too inaccurate. Accordingly, to gauge the accuracy of the model, the proposed algorithm checks the number of vectors inside the TR. This number is then compared to the dimension of the objective function (d), as it is an indicator to the number of function evaluations required to estimate the gradient by finite-differences and hence is indicative of the number of function evaluation needed to find a new optimum. To manage the accuracy of the model, the following steps are performed:
  - If there are less than d SF vectors inside the TR: The unsuccessful step may be since the model or classifier are inaccurate in the TR. As such, the algorithm adds a new point ($\mathbf{x}_\mathrm{n}$) inside the TR to improve their local accuracy. The procedure for adding the point is explained below.
  - If there are more than d SF vectors in the TR: The model and classifier are considered to be sufficiently accurate in the TR. In this case, and following the classical TR framework, the TR is contracted by halving its radius.

Compared to the classical TR framework, the above steps also monitor the number of interior vectors in the TR, since they determine the local model accuracy. Monitoring the number of these vectors ensures the TR is not contracted too quickly when the search stagnates due to poor accuracy of the model or the classifier [3].

Another change from the classical framework is the addition of a new vector ($\mathbf{x}_\mathrm{n}$) to improve the local model accuracy. To accomplish this, the new vector should be far from existing ones, so it improves the model in an region sparse with

TABLE I
EA PARAMETERS

population size	100
generations	100
selection	stochastic universal selection (SUS), $p = 0.7$
recombination	intermediate, $p = 0.7$
mutation[1]	Breeder Genetic Algorithm (BGA) mutation, $p = 0.1$
elitism	10%

[1] Based on [2]

 ISBN: 978-1-61208-154-0

sampled points [13]. Mathematically, finding such a point translates to the following max-min optimization problem,

$$\mathbf{x}_{\mathrm{n}} : \max_{\mathbf{x}\in\mathscr{T}} \min_{\mathbf{x}_i\in\mathscr{T}} \{\|\mathbf{x}-\mathbf{x}_i\|_2\} \tag{15}$$

where $\mathbf{x}_i$, $i = 1\ldots r$, are the existing interior TR points [9]. To simplify the solution of Equation (15), the proposed algorithm generates a LH sample in the TR and chooses the sample point with the largest minimum distance.

Lastly, if the TR has been contracted for q consecutive iterations, which suggests convergence to a local optimum, the algorithm adds a point outside the TR to improve the accuracy of model globally, which assists in locating new optima. The point is generated using the same procedure described above for the new interior point, namely, $\mathbf{x}_{\mathrm{n}}$, but now considering the entire search space instead of just the TR. Based on numerical experiments, we identified $q = 2$ as a suitable setting. To complete the description, Algorithm 1 gives the pseudocode of the proposed algorithm.

Algorithm 1: Proposed Optimization Algorithm with Adaptive Model and Classifier

```
generate an initial LHD sample;
evaluate and cache the sample vectors;
repeat
    train a new Kriging model using all SF vectors in the cache;
    /* classifier selection                                    */
    for candidate classifier = {k-NN, LDA, SVM} do
        use CV to find the classification error (12);
    select the classifier with the lowest error and train a new classifier using all the vectors in the cache;
    /* TR trial step                                           */
    set the best vector in the cache as the TR centre;
    search for the model optimum using an EA and the modified objective function (14);
    evaluate the predicted optimum with the expensive objective function;
    /* manage the model and TR                                 */
    if the new optimum is better than the TR centre then
        increase the TR radius
    else if the new optimum is not better than the TR centre and there are insufficient vectors in the TR then
        add a new vector in the TR to improve the model and classifier;
    else if the new optimum is not better than the TR centre and there are sufficient vectors in the TR then
        decrease the TR radius;
    /* check search stagnation                                 */
    if there have been q consecutive TR contractions then
        add a new vector outside the TR to improve the accuracy of the model globally;
    cache all new vectors evaluated;
until optimization budget exhausted;
```

IV. Performance Analysis

For its evaluation, the proposed algorithm was applied to an engineering application of airfoil shape optimization. The problem is pertinent to this study as it is both representative of real-world expensive black-box optimization problems, *and* contains SI vectors, as explained below.

The setup of the problem is as follows. During flight an aircraft generates *lift*, namely, the beneficial aerodynamic force which keeps it airborne, and also *drag*, that is, an aerodynamic friction force which obstructs the aircraft's movement. Accordingly, the optimization goal is to find an airfoil shape which maximizes the ratio of the lift to drag at some prescribed flight conditions, namely, the flight altitude, the flight speed, and the angle of attack (AOA) which is the angle between the airfoil chord and the aircraft velocity. Figure 2(a) shows the physical quantities involved.

To ensure structural integrity, the minimum airfoil thickness (t) between 0.2 to 0.8 of the airfoil chord must be equal to or larger than a critical value $t^\star = 0.1$. Also, in practise the design requirements for airfoils are specified in terms of the non-dimensional lift and drag coefficients, c_l and c_d, respectively, defined as

$$c_l = \frac{L}{\frac{1}{2}\rho V^2 S} \tag{16a}$$

$$c_d = \frac{D}{\frac{1}{2}\rho V^2 S} \tag{16b}$$

where L and D are the lift and drag forces, respectively, ρ is the air density, V is aircraft speed, and S is a reference area, such as the wing area. Accordingly, maximizing the lift and minimizing the drag is formulated as a minimization problem using the following objective function

$$f = -\frac{c_l}{c_d} + p\,, \tag{17a}$$

where c_l and c_d were defined above, and p is a penalty for airfoils which violate the thickness constraint, and is defined as

$$p = \begin{cases} \dfrac{t^\star}{t}\cdot\left|\dfrac{c_l}{c_d}\right| & \text{if } t < t^\star \\ 0 & \text{otherwise} \end{cases} \tag{17b}$$

Airfoils were represented with the Hicks-Henne parameterization [8], which uses a baseline airfoil and adds the basis functions

$$b_i(x) = \left[\sin\left(\pi x^{\frac{\log(0.5)}{\log(i/(h+1))}}\right)\right]^4, \tag{18}$$

with $i = 1\ldots h$, where h is user-prescribed, to smoothly modify the baseline shape [22]. The lower and upper curves of a candidate airfoil are then given by

$$y = y_b + \sum_{i=1}^{h} \alpha_i b_i(x)\,, \tag{19}$$

where y_b is the baseline upper/lower curve, which was taken as the NACA0012 symmetric airfoil, and $\alpha_i \in [-0.01, 0.01]$ are

the coefficients (design variables) to be found. In this study, we used $h = 10$ functions for the upper and lower curve, respectively, or a total of 20 coefficients, namely, design variables, per airfoil. To obtain the lift and drag of candidate airfoils, we used XFoil–a computational fluid dynamics simulation for analysis of subsonic airfoils [4]. Each airfoil evaluation required up to 30 seconds on a desktop computer. Figure 2(a) gives the layout of the Hicks-Henne parametrization.

As mentioned above, the airfoil optimization problem is a pertinent test case since it contains SI vectors. The prevalence of these vectors depends on two major factors: the AOA *and* the operating conditions, namely, the altitude and velocity. To illustrate the effect of the AOA, 30 different airfoils were evaluated at identical flight conditions, except for the AOA which was increased from 0° to 40°, and the number of failed evaluations, namely, SI vectors, was recorded at each AOA. Figure 2(b) shows the obtained results, which indicate a consistent trend where a higher AOA resulted in more failed evaluations, namely, more SI vectors. Accordingly, we have selected the settings AOA $= 20^\circ, 30^\circ,$ and 40° for the optimization tests. With respect to the altitude and velocity, we have experimented with different operating conditions, and have chosen a speed of $Ma = 0.775$, namely, 0.775 of the speed of sound, and an altitude of 32 kft.

For a comprehensive evaluation, the proposed algorithm was also benchmarked against the following two representative model-assisted EAs:

- *Model-assisted EA with periodic sampling* (EA–PS) [18]: The algorithm begins by generating an initial LH sample and training a Kriging model. A real-coded EA then runs for 10 generations while evaluating only the model, and next, the top 10 elites in the population are evaluated with the true expensive function and are incorporated into the model. The goal of this procedure is to safeguard the accuracy of the model by periodically updating it with the evaluated elites. This optimization loop repeats until the optimization budget is exhausted. In the benchmarks, the EA was identical to the one in the proposed algorithm, and SI vectors were assigned a fictitious penalty taken to be the mean objective value in the initial LH sample.
- *Expected-Improvement with a model-assisted CMA-ES* (EI–CMA-ES) [1]: The algorithm begins by generating an initial sample of points and trains an initial Kriging model. The main loop then begins, where at each generation the algorithm trains a local Kriging model around the current elite using both the recently evaluated vectors, and the cached vectors which are nearest to the elite. The algorithm excludes SI vectors from the model training set. A covariance matrix adaptation evolutionary strategy (CMA-ES) algorithm then searches for an optimum of the model in a bounded region defined by the latter two sets of solutions, namely, the recently evaluated ones and the nearest neighbours, and in the spirit of the Expected-Improvement framework [10], uses the merit function

$$\hat{f}(\mathbf{x}) = m(\mathbf{x}) - \rho\,\zeta(\mathbf{x}), \qquad (20)$$

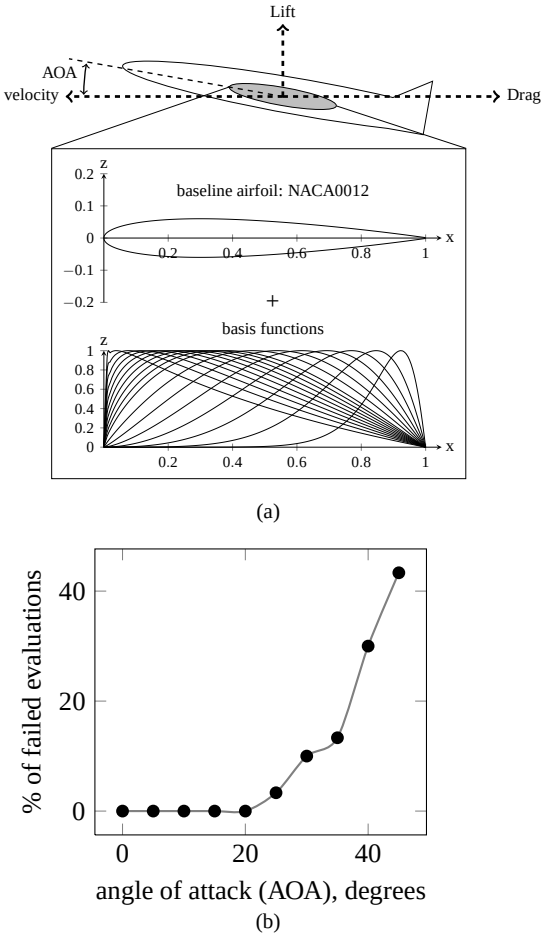


Fig. 2. Aspects of the airfoil optimization problem. (a) shows the physical quantities and Hicks-Henne airfoil parametrization setup. (b) shows the effect of the AOA on the prevalence of SI vectors.

where $m(\mathbf{x})$ is the Kriging model prediction, ρ is a prescribed coefficient, and $\zeta(\mathbf{x})$ is the estimate of the Kriging model prediction error, which is zero at sampled points since there the true objective value is known. The search is repeated for $\rho = 0, 1, 2,$ and 4 to obtain four solutions corresponding to different search profiles, namely, ranging from a local search ($\rho = 0$) to a more explorative one ($\rho = 4$). All non-duplicate solutions found are evaluated with the true expensive function and are cached. In case no new solutions were evaluated, for example, because they already exist in the cache, the algorithm generates a new solution by perturbing the current elite. Following reference [1], the algorithm used a training set of 100 vectors (50 most recently evaluated ones and 50 nearest-neighbours) and the CMA-ES used the default values in the source code [7].

We have also used a variant of the proposed algorithm with a fixed classifier type, namely, only k-NN, to gauge the contribution of the classifier selection step. The variant is designated KK (Kriging–k-NN).

In all tests the optimization budget was 200 evaluations of the true objective function, that is, simulation runs, and the

 ISBN: 978-1-61208-154-0

size of the initial sample was 20. To support a valid statistical analysis, 30 trials were repeated for each algorithm–test case combination.

Table II gives the test statistics for the three AOA cases, as well as the significance-level (α) at which the proposed algorithm was better than each of the other algorithms, namely, EA–PS, EI–CMA-ES, and KK, where an empty entry indicates no statistically-significant difference up to the 0.05 level. Statistical significance tests were done using the Mann–Whitney nonparametric test. For each AOA case, the best mean and median results are emphasized. From studying the test results for each AOA setting it follows:

- AOA=20°: The proposed algorithm obtained the best mean score, and its performance was statistically-significant better than the KK and EA–PS variants at the $\alpha = 0.01$ level. The EI–CMA-ES algorithm had the best median result. followed by the proposed algorithm. With respect to the standard deviation, the EA–PS algorithm had the best (lowest) result, followed by the proposed algorithm.
- AOA=30°: The KK variant obtained the best mean, followed by the proposed algorithm, a setup which was also repeated for the median statistic. The proposed algorithm was statistically-significant better than the EI–CMA-ES algorithm at the 0.01 level. With respect to the standard deviation, the KK algorithm had the best result, followed by the EA–PS algorithm, followed by the proposed algorithm.
- AOA=40°: The proposed algorithm had the best statistic, while the EA–PS algorithm had the best median, closely followed by the proposed algorithm. The proposed algorithm was statistically-significant better than the EI–CMA-ES algorithm at the 0.01 level. With respect to the standard deviation, the KK algorithm had the best result, followed by the proposed algorithm.

Overall, results show the proposed algorithm performed well, as it obtained either the best or near-best mean statistic, and consistently obtained the near-best median statistic, showing its performance was robust across different problem settings. Its standard deviation was intermediate between the extremal results by the other algorithms, indicating there was some variability in its performance, but it was still competitive and never the worst performing algorithm with respect to this statistic. Results also highlight the contribution of the classifier selection stage, as in two cases (AOA $= 20°$ and $40°$) the proposed algorithm outperformed the KK variant which does not select a classifier, and obtained results which were nearly as good in the AOA $= 30°$ case.

Lastly, to demonstrate the optimization outcomes, Figure 3 shows representative airfoils obtained by the proposed algorithm at each of the three optimization cases.

V. SUMMARY

The modern engineering design process is often a simulation-driven optimization problem. In practise, there may exist candidate designs which 'crash' the simulation and thus can lead to search stagnation and to a poor final result. To effectively handle this scenario, this paper has proposed a model-assisted computational intelligence optimization algorithm which introduces a classifier into the search. The latter predicts which solutions are expected to crash the simulation, and its prediction is incorporated with the model prediction to bias the search towards valid solutions, that is, which are expected not to crash the simulator. To improve its efficacy, the proposed algorithm continuously selects during the search an optimal type of classifier, out of a prescribed family of candidates. To safeguard the optimization search in the face of model inaccuracy, the proposed algorithm also employs a TR approach. Performance analysis using a representative real-world application of airfoil shape optimization showed the efficacy of the proposed algorithm. In future work, we consider applying the proposed algorithm to additional challenging real-world applications with SI vectors.

TABLE II
STATISTICS FOR OBJECTIVE VALUE

AOA		P	KK	EA–PS	EI–CMA-ES
20°	mean	**-1.035e+01**	-8.091e+00	-6.889e+00	-1.023e+01
	SD	1.326e+00	1.697e+00	6.526e-01	2.025e+00
	median	-1.049e+01	-7.283e+00	-6.843e+00	**-1.107e+01**
	min	-1.302e+01	-1.138e+01	-8.837e+00	-1.192e+01
	max	-7.143e+00	-5.880e+00	-5.794e+00	-5.442e+00
	α		0.01	0.01	
30°	mean	-3.155e+00	**-3.192e+00**	-3.146e+00	-2.910e+00
	SD	4.694e-02	3.105e-02	3.345e-02	4.761e-02
	median	-3.140e+00	**-3.183e+00**	-3.140e+00	-2.916e+00
	min	-3.270e+00	-3.298e+00	-3.223e+00	-3.005e+00
	max	-3.091e+00	-3.145e+00	-3.092e+00	-2.813e+00
	α				0.01
40°	mean	**-2.793e+00**	-2.784e+00	-2.784e+00	-2.552e+00
	SD	3.380e-02	2.230e-02	4.732e-02	4.249e-02
	median	-2.785e+00	-2.782e+00	**-2.786e+00**	-2.557e+00
	min	-2.875e+00	-2.827e+00	-2.869e+00	-2.637e+00
	max	-2.726e+00	-2.742e+00	-2.717e+00	-2.455e+00
	α				0.01

P: proposed algorithm with model and classifier adaptation.
KK: proposed algorithm restricted to a Kriging model and a *k*-NN classifier.
EA–PS: EA with periodical sampling [18].
EI–CMA-ES: Expected Improvement framework with a CMA-ES optimizer [1].

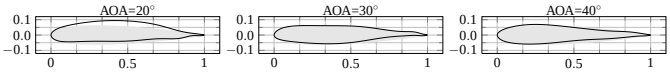


Fig. 3. Representative airfoils obtained by the proposed algorithm, shown in black. For comparison, the baseline NACA0012 airfoil is shown in gray.

ACKNOWLEDGEMENT

The first author thanks the Japan Society for Promotion of Science for its support.

REFERENCES

[1] D. Büche, N. N. Schraudolph, and P. Koumoutsakos. Accelerating evolutionary algorithms with Gaussian process fitness function models. *IEEE Transactions on Systems, Man, and Cybernetics–Part C*, 35(2):183–194, 2005.

[2] A. Chipperfield, P. Fleming, H. Pohlheim, and C. Fonseca. *Genetic Algorithm TOOLBOX For Use with MATLAB, Version 1.2*. Department of Automatic Control and Systems Engineering, University of Sheffield, Sheffield, 1994.

[3] A. R. Conn, N. I. M. Gould, and P. L. Toint. *Trust Region Methods*. SIAM, Philadelphia, Pennsylvania, 2000.

[4] M. Drela and H. Youngren. *XFOIL 6.9 User Primer*. Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, MA, 2001.

[5] M. T. M. Emmerich, A. Giotis, M. Özedmir, T. Bäck, and K. C. Giannakoglou. Metamodel-assisted evolution strategies. In J. J. Merelo Guervós, editor, *The 7th International Conference on Parallel Problem Solving from Nature–PPSN VII*, number 2439 in Lecture Notes in Computer Science, pages 361–370. Springer, Berlin, 2002.

[6] S. Handoko, C. K. Kwoh, and Y.-S. Ong. Feasibility structure modeling: An effective chaperon for constrained memetic algorithms. *IEEE Transactions on Evolutionary Computation*, 14(5):740–758, 2010.

[7] N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, 2001. http://www.lri.fr/~hansen/cmaes_inmatlab.html.

[8] R. M. Hicks and P. A. Henne. Wing design by numerical optimization. *Journal of Aircraft*, 15(7):407–412, 1978.

[9] M. E. Johnson, L. M. Moore, and D. Ylvisaker. Minimax and maximin distance designs. *Journal of Statistical Planning and Inference*, 26(2): 131–148, 1990.

[10] D. R. Jones, M. Schonlau, and W. J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13: 455–492, 1998.

[11] J. R. Koehler and A. B. Owen. Computer experiments. In S. Ghosh, C. R. Rao, and P. R. Krishnaiah, editors, *Handbook of Statistics*, pages 261–308. Elsevier, Amsterdam, 1996.

[12] K.-H. Liang, X. Yao, and C. Newton. Evolutionary search of approximated N-dimensional landscapes. *International Journal of Knowledge-Based Intelligent Engineering Systems*, 4(3):172–183, 2000.

[13] W. R. Madych. Miscellaneous error bounds for multiquadric and related interpolators. *Computers and Mathematics with Applications*, 24(12): 121–138, 1992.

[14] M. D. McKay, R. J. Beckman, and W. J. Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21(2):239–245, 1979.

[15] T. Okabe. Stabilizing parallel computation for evolutionary algorithms on real-world applications. In *Proceedings of the 7th International Conference on Optimization Techniques and Applications–ICOTA 7*, pages 131–132. Universal Academy Press, Tokyo, 2007.

[16] C. Poloni, A. Giurgevich, L. Onseti, and V. Pediroda. Hybridization of a multi-objective genetic algorithm, a neural network and a classical optimizer for a complex design problem in fluid dynamics. *Computer Methods in Applied Mechanics and Engineering*, 186(2–4):403–420, 2000.

[17] K. Rasheed, H. Hirsh, and A. Gelsey. A genetic algorithm for continuous design space search. *Artificial Intelligence in Engineering*, 11:295–305, 1997.

[18] A. Ratle. Optimal sampling strategies for learning a fitness model. In *The 1999 IEEE Congress on Evolutionary Computation–CEC 1999*, pages 2078–2085. IEEE, Piscataway, New Jersey, 1999.

[19] Y. Tenne and S. W. Armfield. A versatile surrogate-assisted memetic algorithm for optimization of computationally expensive functions and its engineering applications. In A. Yang, Y. Shan, and L. Thu Bui, editors, *Success in Evolutionary Computation*, volume 92 of *Studies in Computational Intelligence*, pages 43–72. Springer-Verlag, Berlin; Heidelberg, 2008.

[20] Y. Tenne and C. K. Goh, editors. *Computational Intelligence in Expensive Optimization Problems*, volume 2 of *Evolutionary Learning and Optimization*. Springer, 2010. URL `http://www.springerlink.com/content/v81864`.

[21] Y. Tenne, K. Izui, and S. Nishiwaki. Handling undefined vectors in expensive optimization problems. In C. Di Chio, editor, *Proceedings of the 2010 EvoStar Conference*, volume 6024/2010 of *Lecture Notes in Computer Science*, pages 582–591. Springer, Berlin, 2010.

[22] H.-Y. Wu, S. Yang, F. Liu, and H.-M. Tsai. Comparison of three geometric representations of airfoils for aerodynamic optimization. In *Proceedings of the 16th AIAA Computational Fluid Dynamics Conference*. American Institute of Aeronautics and Astronautics, 2003. AIAA 2003-4095.

[23] X. Wu, V. Kumar, R. J. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, P. S. Yu, Z.-H. Zhou, M. Steinbach, D. J. Hand, and D. Steinberg. Top 10 algorithms in data mining. *Knowledge and Information Systems*, 14:1–37, 2008.

 ISBN: 978-1-61208-154-0