FUTURE COMPUTING 2016

The Eighth International Conference on Future Computational Technologies and Applications

ISBN: 978-1-61208-461-9

March 20 - 24, 2016

Rome, Italy

FUTURE COMPUTING 2016 Editors

Rudolf Berrendorf, Bonn-Rhein-Sieg University, Germany

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

FUTURE COMPUTING 2016

Forward

The Eighth International Conference on Future Computational Technologies and Applications (FUTURE COMPUTING 2016), held between March 20-24, 2016 in Rome, Italy, continued a series of events targeting advanced computational paradigms and their applications. The target 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.

The conference had the following tracks:

- Computing technologies
- Computational intelligence strategies
- Large-scale computing strategies

Similar to the previous edition, this event attracted excellent contributions and active participation from all over the world. We were very pleased to receive top quality contributions.

We take here the opportunity to warmly thank all the members of the FUTURE COMPUTING 2016 technical program committee, as well as the numerous reviewers. The creation of such a high quality conference program would not have been possible without their involvement. We also kindly thank all the authors that dedicated much of their time and effort to contribute to FUTURE COMPUTING 2016. 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 also gratefully thank the members of the FUTURE COMPUTING 2016 organizing committee for their help in handling the logistics and for their work that made this professional meeting a success.

We hope FUTURE COMPUTING 2016 was a successful international forum for the exchange of ideas and results between academia and industry and to promote further progress in the area of future computational technologies and applications. We also hope that Rome provided a pleasant environment during the conference and everyone saved some time for exploring this beautiful city.

FUTURE COMPUTING 2016 Chairs

FUTURE COMPUTING 2016 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, SRH University Berlin - Institute of Information Systems, Germany
Wail Mardini, Jordan University of Science and Technology, Jordan
Alexander Gegov, University of Portsmouth, UK

FUTURE COMPUTING 2016 Industry/Research

Francesc Guim, Intel Corporation, Spain
Wolfgang Gentzsch, The UberCloud, Germany
Noboru Tanabe, Toshiba Corporation, Japan

FUTURE COMPUTING 2016

Committee

Table of Contents

Multiobjective Power Loss Optimization Versus System Stability Assessment for Hydrocarbon Industrial Plant Using Differential Evolution Algorithm

M. T. Al-Hajri
Computer & Electronic Eng. Dept.
Brunel University
Uxbridge, United Kingdom
e-mail:muhammad.al-hajri@brunel.ac.uk

M. A. Abido
Electrical Engineering Department.
King Fahad University (KFUPM)
Dhahran, Kingdom of Saudi Arabia
e-mail:mabido@kfupm.edu.sa

M. K. Darwish
Computer & Electronic Eng. Dept.
Brunel University, U.K.
Uxbridge, United Kingdom
e-mail:mohamed.darwish@brunel.ac.uk

***Abstract*- In this paper, a differential evolution algorithm (DEA) is considered for formulating nonlinear constrained multiobjective problem where electrical system real power loss and voltage stability index are optimized concurrently for a real hydrocarbon industrial plant. The subject plant electrical system consists of 275 buses, two gas turbine generators, two steam turbine generators, large synchronous motors, and other rotational and static loads. Truncation mechanism is used to manage the pareto-optimal solution set size. The best compromise solution is extracted using fuzzy set theory. The DEA performance for different population sizes and generation number cases will be demonstrated. The results exhibited the capabilities of the proposed approach in producing optimized pareto-optimal solutions for the subject multiobjective problem. The byproduct annual cost avoidance potential due to real power loss optimization for the studied cases will be demonstrated.**

Keywords-differential evolution algorithm; power loss optimization; voltage stability index; hydrocarbon facility; millions of standard cubical feet of gas (MMscf).

I. INTRODUCTION

Due to the exponential increase of the electrical power demand and the average low generation efficiency in most of the developing countries, the issue of reducing the electrical power system loss while maintaining the system stability has received more attention. For example, in Saudi Arabia, the annual average peak electric demand increase is around 7.4% [1]. As of 2013, the distribution of plant capacity for electricity generation in Saudi Arabia, by technology is shown in Figure 1. The low efficient simple cycle steam turbine generation is making 32% of Saudi Arabia utility company generation fleet while the most efficient combined cycle is around 13.8% of the whole fleet [2]. These inspired most of the developing countries to unleash national initiatives to optimize electrical usage and reduce system loss. The subject issue can be addressed by adjusting transformer taps and generators and synchronous motors buses voltages.

Heuristic methods are very powerful in addressing the electricals system loss optimization by searching the solution space for the optimal solutions. Many intelligent algorithms were implemented to identify the pareto-optimal solution set for the subject. The DEA in most cases demonstrates superior

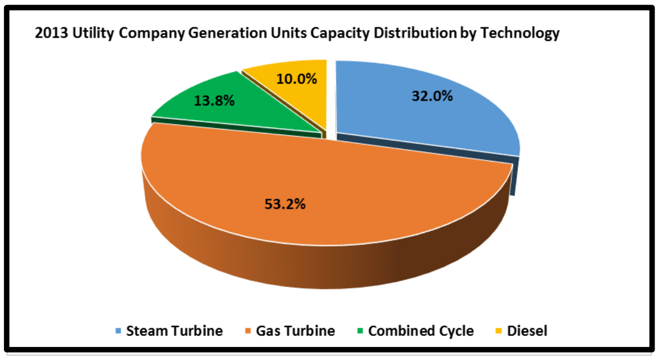


Figure 1. Saudi Arabia generation units fleet distribution by technology

over other algorithms such as the genetic algorithm [3]-[6]. Most of the previous studies in the literature used virtual IEEE system models to address the power loss and system stability multiobjective optimization problem or other problems [7]-[9]. A truncation technique is implemented with the DEA to manage the size of the Pareto-optimal set solution [9]. The fuzzy logic is the most common technique in extracting the best compromise solution out of the pareto-optimal solution set [7][9].

This paper considers an existing real life hydrocarbon central processing facility electrical power system model for assessing the potential of DEA in optimizing two competing objectives simultaneously: power system loss and system stability index improvement. In Section 2 of the paper, the problem will be formulated as a multiobjective optimization problem with equality and inequality constraints. In Section 3, the multiobjective optimization process will be illustrated. In Section 4, the DEA approach will be addressed. In Section 5, the paper study cases will be developed. Finally, in Section 6 the technical and economic analysis of the studied cases results will be presented.

II. PROBLEM FORMULATION

The problem formulation consists of five parts: the development of the objective functions, the calculation of all load buses stability index (L-Index), the identification of the system electrical constraints to be met - equality and inequality constraints - and the illustration of the fuzzy logic and the truncation technique.

A. Problem Objective Functions

In this paper, two competing objective functions will be addressed as follows:

A.1 System Loss Objective Function

This objective function is to minimize the real power loss J_1 (P_{Loss}) in the transmission and distribution lines. This objective function can be expressed in term of the power follow loss between two buses *i* and *j* as follows:

$$J_1 = \mathbf{P_{Loss}} = \sum_{k=1}^{nl} g_k \, [V_i^2 + V_j^2 - 2\, V_i V_j \cos(\delta_i - \delta_i)] \qquad (1)$$

where *nl* is the number of transmission and distribution lines; g_k is the conductance of the k^{th} line, $V_i \angle\delta_i$ and $V_j \angle\delta_j$ are the voltage at end buses *i* and *j* of the k^{th} line, respectively [7] [9].

The objective is to minimize P_{loss}, that is,
J_1= Minimize (P_{loss}) (2)

A.2 Voltage Stability Index (L_{max})

The *L* indicator varies in the range between 0 (the no load case) and 1, which corresponds to voltage collapse. This indicator uses the bus voltage and network information provided by the power flow program to measure the stability of the system. The *L* indicator can be calculated as given in [10]. For a multi-node system

$$I_{bus} = Y_{bus} \times V_{bus} \qquad (3)$$

By segregating the load buses (PQ) from generator buses (PV), (3) can be written as:

$$\begin{bmatrix} I_L \\ I_G \end{bmatrix} = \begin{bmatrix} Y_1 & Y_2 \\ Y_3 & Y_4 \end{bmatrix} \begin{bmatrix} V_L \\ V_G \end{bmatrix} \qquad (4)$$

$$\begin{bmatrix} V_L \\ I_G \end{bmatrix} = \begin{bmatrix} H_1 & H_2 \\ H_3 & H_4 \end{bmatrix} \begin{bmatrix} I_L \\ V_G \end{bmatrix} = \begin{bmatrix} Z_{LL} & F_{LG} \\ K_{GL} & Y_{GG} \end{bmatrix} \begin{bmatrix} I_L \\ V_G \end{bmatrix} \qquad (5)$$

where
V_L , I_L *are load buses voltages and currents*
V_G , I_G *are Generator buses voltages and currents*
H_1 ,H_2 ,H_3 ,H_4 *are submatrices generated from* Y_{bus} *Partial Inversion*
Z_{LL} ,F_{LG} , K_{GL} ,Y_{GG} *are submatrices of H-matrix*

Therefore, a local indicator L_j can be worked out for each node j similar to the line model

$$L_j = \left| 1 - \frac{\sum_{i \in \alpha G} F_{ji} V_i}{V_j} \right| \qquad (6)$$

For a stable situation, the condition $L_j \leq 1$ must not be violated for any of the nodes j. Therefore, a global indicator *L* describing the stability of the whole system is given by:

$$L_{max} = \mathrm{MAX}_{j \in \alpha L} \left| 1 - \frac{\sum_{i \in \alpha G} F_{ji} V_i}{V_j} \right| \qquad (7)$$

where α_L is the set of load buses and α_G is the set of generator buses.

The objective is to minimize L_{max}, that is,
J_2= Minimize (L_{max}) (8)

Combining the objectives functions and these constraints, the problem can be mathematically formulated as a nonlinear constrained single objective optimization problem as follows:

Minimize J_1 and J_2
Subject to:

$$g(x,u) = 0 \qquad (9)$$
$$|h(x,u)| \leq 0 \qquad (10)$$

where:
x: is the vector of dependent variables consisting of load bus voltage V_L, generator reactive power outputs Q_G and the Synchronous motors reactive Power Q_{Synch}. As a result, x can be expressed as

$$x^T = [V_{L1}..V_{LNL}, Q_{Gi}\ldots Q_{GNG}, Q_{Synchi}\ldots Q_{SynchNSynch}] \qquad (11)$$

u: is the vector of control variables consisting of generator voltages V_G, transformer tap settings T, and synchronous motors voltage V_{Synch}. As a result, u can be expressed as

$$u^T = [V_{G1}..V_{GNL}, T_1\ldots T_{NT}, V_{Synch1}..V_{SynchNL}] \qquad (12)$$

g: are the equality constraints.
h: are the inequality constraints.

B. Problem Equality and Inequality Constraints

The system constraints are divided into two categories: equality constraints and inequality constraints [6][7]. Details are as follows:

B.1 Equality Constraints

These constraints represent the power load flow equations. The balance between the active power injected P_{Gi}, the active power demand P_{Di} and the active power loss P_{li} at any bus i is equal to zero. The same balance apply for the reactive power Q_{Gi}, Q_{Di} , and Q_{li}. These balances are presented as follows:

$$P_{Gi} - P_{Di} - P_{li} = 0 \qquad (13)$$

$$Q_{Gi} - Q_{Di} - Q_{li} = 0 \qquad (14)$$

The above equations cane be detailed as follows:

$$P_{Gi} - P_{Di} - V_i \sum_{j=1}^{NB} \boldsymbol{V_j} \, [\, G_{ij} cos(\delta_i - \delta_j) + B_{ij} \sin(\delta_i - \delta_j)] = 0 \qquad (15)$$

$$Q_{Gi} - Q_{Di} - V_i \sum_{j=1}^{NB} \boldsymbol{V_j} \, [\, G_{ij} sin(\delta_i - \delta_j) - B_{ij} \cos(\delta_i - \delta_j)] = 0 \qquad (16)$$

where i = 1,2,…,NB;NB is the number of buses; P_G and Q_G are the generator real and reactive power, respectively; P_D and Q_D are the load real and reactive power, respectively; G_{ij} and B_{ij} are the conductance and susceptance between bus i and bus j, respectively.

B.2 Inequality Constraints

These constraints represent the system operating constraints posted in Table IV.

C. Fuzzy Logic for Selecting the Best Compromise Solution

Upon having the Pareto-optimal set of nondominated solution, the proposed approach presents one solution to the decision maker as the best compromise solution. Due to imprecise nature of the decision maker's judgment, the *i*-th objective function F_i is represented by a membership function μ_i defined as [7] [9].

$$\mu_i = \begin{cases} 1 & F_i \le F_i^{\min} \\ \dfrac{F_i^{\max} - F_i}{F_i^{\max} - F_i^{\min}} & F_i^{\min} < F_i < F_i^{\max} \\ 0 & F_i \ge F_i^{\max} \end{cases} \qquad (17)$$

where $F_i^{\min}$ and $F_i^{\max}$ are the minimum and maximum value of the *i*-th objective function among all nondominated solutions, respectively.

For each nondominated solution *k*, the normalized membership function μ^k is calculated as

$$\mu^k = \frac{\sum_{i=1}^{N_{obj}} \mu_i^k}{\sum_{k=1}^{M} \sum_{i=1}^{N_{obj}} \mu_i^k} \qquad (18)$$

where *M* is the number of nondominated solutions. The best compromise solution is that having the maximum value of μ^k.

D. Pareto Set Reduction by Truncation

A minimum distance based algorithm [9] is employed to reduce the Pareto set to manageable size. At each iteration, an individual *i* is chosen for removal from the external pareto set P_{t+1}. The algorithm is illustrated in the following steps:

Step 1: find the nearest individuals A and B in the objective functions space.

Step 2: calculate the distance d_A and d_B of the next nearest individual from A and B in the objective function space respectively.

Step 3: Delete the individual with the smaller one between d_A and d_B.

III. MULTIOBJECTIVE OPTIMIZATION

In real life, there are problems that involve simultaneous optimization with no common thing in-between. Usually, these problems are competing multiobjective optimization which mandates developing a set of optimal solutions, instead of one optimal solution. The reason for the optimality of many solutions is that no one can be considered to be better than any other with respect to all objective functions. These optimal solutions are known as Pareto- optimal solutions set.

A universal multiobjective optimization problem consists of a number of objectives to be optimized simultaneously in association with a number of equality and inequality constraints. It can be formulated as follows:

$$Minimize \;\; f_i(x) \;\; i = 1,..., N_{obj} \qquad (19)$$

$$Subject\ to: \begin{cases} g_j(x) = 0 & j = 1,..., M \\ |\, h_k(x) | \le 0 & k = 1,..., K \end{cases} \quad \text{Constraints} \qquad (20)$$

where f_i is the i^{th} objective functions, *x* is a decision vector that represents a solution, and N_{obj} is the number of objectives.

For a multiobjective optimization problem, any two solutions x^1 and x^2 can have one of two possibilities: one covers or dominates the other or none dominates the other. In a minimization problem, without loss of generality, a solution x^1 dominates x^2 if the following two conditions are satisfied:

$$1.\ \forall i \in \{1, 2, ..., N_{obj}\}: \; f_i(x^1) \le f_i(x^2) \qquad (21)$$

$$2.\ \exists j \in \{1, 2, ..., N_{obj}\}: \; f_j(x^1) < f_j(x^2) \qquad (22)$$

If any of the above conditions is violated, solution x^1 does not dominate the solution x^2. If x^1 dominates solution x^2, x^1 is called the nondominated solution. The solutions that are nondominated within the entire search space are denoted as *Pareto-optimal* and constitute the *Pareto-optimal set*.

IV. THE DEA APPROACH

The DEA evolution process are summarized in the following steps [11] [12]:

Step 1: Initialization

The population P_0 is generated with *K* size and an vacant annals (external) Pareto-optimal set $\overline{P}_0$ with $\overline{K}$ size.

Step 2: Updating of external pareto set

To bring the external pareto-optimal set up to date, the following steps are to be shadowed,

(a) Population non-dominated individuals are highlighted and reproduce to the external Pareto set.
(b) Look for set of external Pareto, designed for the non-dominated individuals.
(c) If condition ($\overline{P_{t+1}}$) < $\overline{K}$ is satisfied, keep the individuals with higher fitness values untial | $\overline{P_{t+1}}$| = $\overline{K}$ is satisfied.
(d) If ($\overline{P_{t+1}}$) > $\overline{K}$, truncation procedure is called which removes individuals from ($\overline{P_{t+1}}$) in anticipation of | $\overline{P_{t+1}}$| = $\overline{K}$.

Step 3: Assignment of fitness values

The fitness values of the individuals are calculated in the external Pareto set $\overline{P_t}$ and the population P_t *as* follows:

(a) St(*i*); strength value; is assigned to all individuals *i* inside the external pareto set $\overline{P_t}$ and the population P_t. St(*i)* signifies the unit, which *i* dominates and it is expressed as follow:

$$St(i) = |\{ j, j \in P_t + \overline{P}_t, \wedge i \succ j \}| \tag{23}$$

Then, the raw fitness $R_w(i)$ with respect to an individual can be measured as follows:

$$R_w(i) = \sum_{j \epsilon P_t + \overline{P}_t, j \succ i} St(j) \tag{24}$$

The raw fitness of an individual is obtained with respect to the strength of its dominators in the archive and population.

(b) The distances between an individual *i* and the entire *j* individuals, in the course of external and population sets and are enlisted. Then, the list is sorted in a cumulative manner, the distance to the *m*th individual, consequently $m = \sqrt{\overline{K} + K}$ is represented as σ_i^m. Then, the density D(*i*) is calculated for each *i*

$$D(i) = \frac{1}{\sigma_i^m + 2} \tag{25}$$

The addition of integer 2 is made in the denominator to certify that the value of D (*i*) is larger than zero and is < 1. The fitness value *i* of an individual *is* expressed as follows*:*

$$F(i) = R_w(i) + D(i) \tag{26}$$

Step 4: Mutation

Different from the SPEA2 in which the individuals to be subjected to crossover and mutation are selected from the front pareto optimal set, in DEA the individuals are selected from the population. In the DEA, mutation is performed using the DE/rand/1 mutation technique. $V_i(t)$, the mutated vector, is created for each population member $X_i(t)$ set by randomly selecting three individuals' x_{r1}, x_{r2} and x_{r3} and not corresponding to the current individual x_i. Then, a scalar number F is used to scale the different between any two of the selected individuals. The resultant difference is added to the third selected individual. The mutation process can be written as:

$$V_{i,j}(t) = x_{r1,j}(t) + F \cdot [x_{r2,j}(t) - x_{r3,j}(t)] \tag{27}$$

The value of *F* is usually selected between 0.4 and 1.0; in this study F was set to be 0.5 (50%).

Step 5: Crossover

Perform the binomial crossover, which can be expressed as follow:

$$u_{i,j}(t) = \begin{cases} v_{i,j}(t) \text{ } if \text{ } rand \text{ } (0,1) < CR \\ x_{i,j}(t) \qquad\qquad else \end{cases} \tag{28}$$

CR is the crossover control parameter and it usually set within the range [0, 1]. The child $u_{i,j}(t)$ will contend with its parent $x_{i,j}(t)$. CR is set equal to 0.9 (90%) in ths study

Step 6: Selection

The procedure for the selection is as follows:

$$x_i(t+1) = u_i(t) \quad \text{condition} \quad f(u_i(t)) \leq f(x_i(t)) \tag{29}$$

$$x_i(t+1) = x_i(t) \quad \text{condition} \quad f(x_i(t)) \leq f(u_i(t)) \tag{30}$$

where *f*( ) is the objective function to be minimized.

Step 7: Looping back/Termination

Look for the terminating criteria. If the criteria are not fulfilled, then generate new offspring population to previous one. If satisfied, apply the fuzzy set theory for the identification of the best compromise pareto set. Figure 2. demonstrates DEA evolutionary steps.

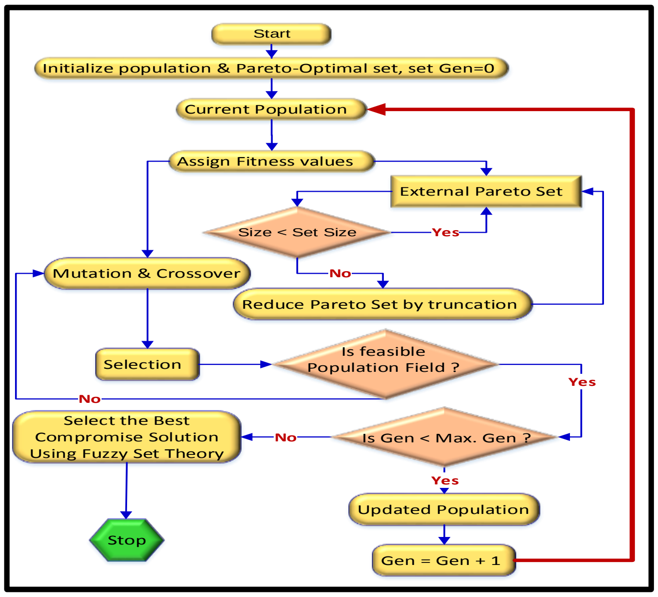


Figure 2. DEA evolutionary process chart

V. STUDY CASES

In this paper, three cases were studied. First, is the base case - business as usual (BAU). Second, the different number of generation with fix population size optimization case.

Third, the different population size with fix number of generation optimization case. In the two optimization cases, the best compromised values of the objective functions (J_1 and J_2) are obtained.

A. Base Case Scenario (Business as Usual)

The base case scenario (BAU) which is also called normal system operation mode was simulated to be benchmarked with the two optimal cases. Following are some of the normal system operation mode parameters:

1) The utility bus and generators terminal buses were set at unity p.u. voltage.
2) All the synchronous motors were set to operate very close to the unity power factor.
3) All downstream distribution transformers' and the captive synchronous motors transformers' off-load tap changers were put on the neutral tap.
4) The causeway substations main transformers' taps were raised to meet the very conservative buses voltage constraints ($\geq$ 0.95p.u.) at these substations downstream buses as posted in Table IV. These main transformers' selected taps values are as per Table I below.

TABLE I
THE SELECTED FEASIBLE TRANSFORMERS TAPS VALUE

Substation Number	Transformer Tap
Causeway Substation#1	+3 (1.019 p.u.)
Causeway Substation#2	Neutral (1.0 p.u.)
Causeway Substation#3	+3 (1.019 p.u.)
Main Substation Transformers	+1 (1.006 p.u.)

B. Different Generation Number with Fix Population Size Case

In this case, the crossover rate is fixed at 90%, the mutation rate is fixed at 50%, the population size was fixed at 100 individuals and the front pareto set population is fixed to be 50 individuals. Yet, the number of generations was varied to be 50, 100 and 150 respectively. The effect of different number of generations on the DEA performance compared to the BAU case is posted in Table II.

TABLE II
EFFECT OF GENERATION NUMBER ON THE DEA PERFORMANCE

Generation #	BAU J_1/J_2	Optimal J_1/J_2	Δ% J_1/J_2
50	2.13/0.075	1.90/0.06636	-10.8%/-11.5%
100	2.13/0.075	1.893/0.06640	-11.1%/-11.5%
150	2.13/0.075	1.897/0.06633	-10.9%/-11.6%

C. Different Population Size with Fix Generation Number Case

In this case, the number of generations is fixed at 50, the crossover rate is fixed at 90%, the mutation rate is fixed at 50% and the front pareto set population is fixed to be 50 individuals. Yet, the population size was varied to be 100, 200 and 300 correspondingly. Table III shows the effect of population size on the DEA performance compared to the BAU case. Population size of 200 produces better optimized value of J_1 while population size of 300 produces better optimized value of J_2.

TABLE III
EFFECT OF POPULATION SIZE ON THE DEA PERFORMANCE

Population Size	BAU J_1/J_2	Optimal J_1/J_2	Δ% J_1/J_2
100	2.13/0.075	1.902/0.06636	-10.8% /-11.5%
200	2.13/0.075	1.892/0.06640	-11.2% /-11.5%
300	2.13/0.075	1.91/0.06633	-10.3% /-11.6%

VI. RESULTS AND DISCUSSIONS

The results from the three studied cases will be analyzed in two categories: the system parameters analysis and the economic analysis.

A. System Parameters Analysis

The hydrocarbon facility simplified electrical system model, which is studied in this paper, is shown in Figure 3.

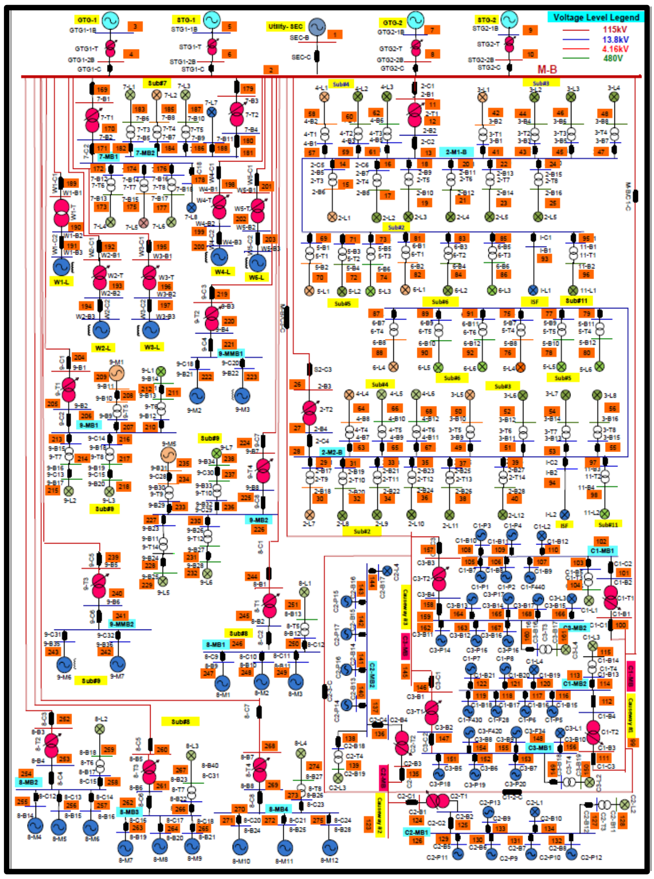


Figure 3. Simplified electrical system of the hydrocarbon facilty

The system inequality constraints are posted in Table IV. All these constraints are real system constraints for industrial electricals system. The real and reactive power limitations are manufactures actual limitations.

TABLE IV
SYSTEM INEQUALITY CONSTRAINTS

Description	Lower Limit	Upper Limit
GTG Terminal Voltage (V_{GTG})	90%	105%
STG Terminal Voltage (V_{STG})	90%	105%
GTG Reactive Power (Q_{GTG}) Limit	-62.12 MVAR	95.72 MVAR
STG-1 Reactive Power (Q_{STG}) Limit	-22.4 MVAR	20.92 MVAR
STG-2 Reactive Power (Q_{STG}) Limit	-41.9 MVAR	53.837 MVAR
Captive Synch. Motors Terminal Voltage	90%	105%
Synch. Motors Terminal Voltage (V_{Sychn})	90%	105%
Causeway downstream Buses Voltage	95%	105%
All Load Buses Voltage	90%	105%
Main Transformer Taps	+16 (+10%)	-16 (-10%)
Generators Step-Up Transformer Taps	+8 (+10%)	-8 (-10%)

The front pareto-optimal solution set for the first optimal studied case - different number of generation with fixed number of generation - is captured in Figure 4.

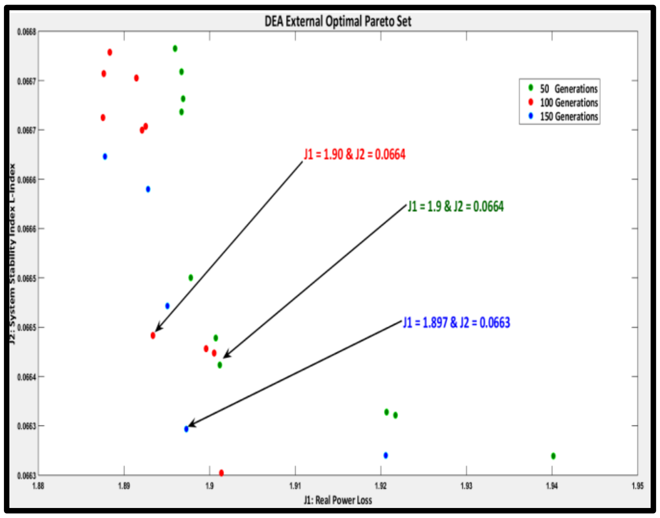


Figure 4. Pareto-optimal set solution for the first optimal case

The second optimal case - different population size - front pareto-optimal solution sets are shown in Figure 5. As shown in both figures none of the different population sizes or generation number produces very well distributed front pareto optimal set. A better distributed front pareto optimal set may be produced by trying higher population size or generation number. Yet, this will increase the evolution process. Massaging the crossover and mutation rate may also results in well distributed front pareto optimal set.

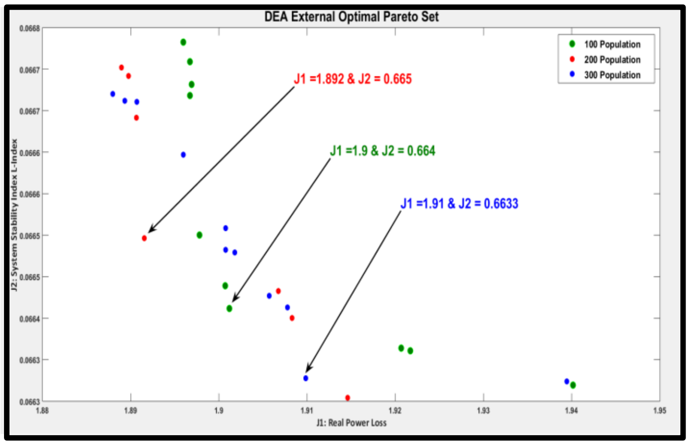


Figure 5. Pareto-optimal set solution for the second optimal case

B. Economic Analysis

The avoided annual cost due to the optimization of the system power loss is demonstrated in Figure 6 for the BAU and the two optimal cases. The annual cost avoidance based on natural gas cost of \$3.5 per MMscf is around \$69,507/year for the 150 generations with 100 population size scenario part of the first optimal case. It is \$71,218/year for 200 population size with 50 generations scenario part of the second optimal case.

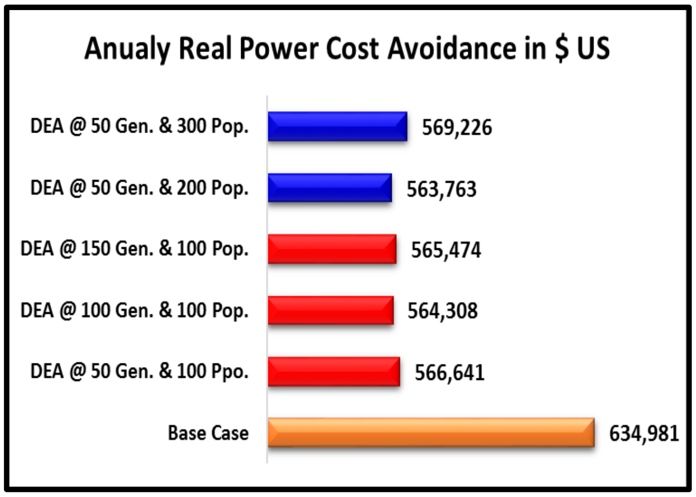


Figure 6. The avoided cost due to system power loss optimization

VII. CONCLUSION AND FUTURE WORK

This paper presented the potential of DEA in addressing the studied multiobjective problem for a real-life hydrocarbon facility. It was clearly demonstrated that increasing of generations number have better impact in producing better pareto-optimal set solution. Yet, many of the pareto-optimal solution set converged to the same J_1 and J_2 values. The annual avoided cost due to the power loss reduction was captured. Future study may need to address the effectiveness of different crossover and mutation rate percentage in the pareto-optimal solution set values and distribution. The use of different fix or dynamic mutation and crossover rate is a subject of future study considering the problem in hand.

ACKNOWLEDGMENTS

The authors acknowledge the support of the Power System Planning Department/Saudi Aramco Oil Company, Brunel University and King Fahd University of Petroleum & Minerals for their support and encouragement throughout the study.

REFERENCES

[1] The Electricity Co-Generation Regulatory Authority (ECRA), “Annual Peak Load for the Kingdom”, 2015, http://ecra.gov.sa/peak_load.aspx#.VOg-5I05BKA, [retrieved: January, 2016].

[2] ‘Saudi Electrical Company 2013 annual report’, https://www.se.com.sa/en-us/Lists/AnnualReports/Attachments/11/AnnualReport2013En.pdf [retrieved: January, 2016].

[3] K. Iba, "Reactive Power Optimization by Genetic Algorithm," IEEE Trans. on Power Systems, Vol. 9, No. 2, 1994, pp. 685-692.

[4] D. Gan, Z. Qu, and H. Cai, "Large-Scale VAR Optimization and Planning by Tabu Search," Electric Power Systems Research, 39, 1996, pp. 195-204.

[5] Y. T. Hsiao and H. D. Chiang, "Applying Network Window Schema and a Simulated Annealing Technique to Optimal VAR Planning in Large-Scale Power systems," Electric Power Systems Research, 22, 2000, pp. 1-8.

[6] P. Yonghong and L. Yi, “An Improved Genetic Algorithm for Reactive Power Optimization,” Control Conference (CCC), 2011 30th Chinese, pp.2105-2109

[7] M. A. Abido, ” Multiobjective Optimal VAR Dispatch Using Strength Pareto Evolutionary Algorithm,” 2006 IEEE Congress on Evolutionary Computation, Vancouver, BC, Canada, July 16-21, 2006, pp. 730-736.

[8] D. Guan, Z. Cai and Z. Kong, “Reactive Power and Voltage Control Using Micro-Genetic Algorithm,” International Conference on Mechatronics and Automation, Changchun, August 9-12, 2009, pp.5019-5024.

[9] Muhammad T. Al-Hajr and M. A. Abido, “Multiobjective optimal power flow using improved strength pareto evolutionary algorithm (SPEA2),” IEEE, 11th International Conference of Intelligent System Design and Application (ISDA), Cordoba, Spain, November 22-24, 2011, pp. 1–7.

[10] C. Belhadj and M. A. Abido, “ An Optimized Fast Voltage Stability Indicator,” IEEE Budapest Power Tech ’99 Conference, Budapest, Hungary, August 29-September 2, 1999, BPT99-363-12 J, pp.79.

[11] M. Varadarajan and K. S. Swarup, “Solving multi-Objective optimal power flow using differential evolution,” IET Generation, Transmission & Distribution, 2008, vol. 2, No. 5, pp. 720–730.

[12] Zhang Xuexia, Chen Weirong and P. N. Suganthan, “Optimal multi-objective reactive power dispatch considering static voltage stability based on dynamic multi-group self-adaptive differential evolution algorithm,” IEEE International Conference on Intelligent system design and engineering application, 2012, , pp. 1448–1456.

SEEV-Effort - Is it Enough to Model Human Attentional Behavior in Public Display Settings?

Benedikt Gollan

Pervasive Computing Applications
Research Studios Austria FG,
Thurngasse 8/20, 1090, Vienna, Austria
Email: benedikt.gollan@researchstudio.at

Alois Ferscha

Institute for Pervasive Computing
Johannes Kepler University,
Altenberger Strae 69, 4040 Linz
Email: ferscha@pervasive.jku.at

***Abstract*—Due to ever-increasing information overload, human attention has evolved to the most critical parameter in the successful design of pervasive display systems. This article aims at the validation of physical effort as a suitable descriptor for attentional and perceptional behavior in interactive public display scenarios. For this purpose, we integrated our qualitative effort-based behavior description approach into Wickens' established Saliency-Effort-Expectancy-Value (SEEV) attention model to compare predicted and observed attentional behavior and demonstrate its significance for attention mechanisms. The SEEV attention model is adapted to a public display scenario, with analysis focused on Minimum Required Effort (MRE) for the assessment of information. This attention modeling approach is evaluated based on data collected in an empiric study in an exhibition setting and is based on a database of 188 visitors. We employ different approaches of correlation analysis to evaluate the dependencies between required effort for information assessment and overt attention behavior which results in a maximum overall correlation score of 0.749 in a frame by frame correlation analysis.**

***Keywords*–attention; behavior analysis; public displays; implicit interaction**

I. Introduction

Human attention has revealed as being among the most crucial, yet least understood design criteria for modern, successful Information and Communication Technologies (ICT) systems. Digital information society technologies like the WWW, social networks or mail and messaging systems continuously wash floods of information to individuals via personal (mobile computers, smartphones) and public ICT systems (public displays, digital signage) [1], making it difficult for the individual to allocate attention to the right things at the right time and filter out the gold nuggets of information, which are relevant in the respective situation and context.

Given the overabundance of information approaching individuals, it has to be of immediate interest to include a profound understanding of (i) how attention is allocated and how it can be, (ii) if not explicitly measured, at least be estimated and modeled into technical solutions in public display scenarios. Only a fundamental understanding and successful modeling of human attention processes will allow the development of pervasive ICT systems that are designed for the benefit of users, and not only aiming at a most efficient distribution of arbitrary information to a maximum audience.

In the following, the current state of the art will be addressed. In Section 2, the underlying SEEV attention model and necessary simplifications for this work are described. In Section 3, the underlying experiment is introduced based on which the behavior data computation is carried out (Section 4). Results are presented and discussed in Section 5, to be concluded a discussion in Section 6.

A. Related Work

Attention research has evolved from first general observations by James [2] in the 1880s, over different Single Channel Theories [3–7] in which mental processes are regarded as serial competitive activities, over Capacity Theories in which human attention is not limited by perception bandwidth or number of channels but by processing capacity [8–10] to nowadays multi-channel and multiple-resource theories [11, 12]. These theories include multitasking capabilities and at the same time investigate distinct aspects of attention in detail. All these single models contribute to the overall understanding of the complex matter of human attention, yet, the latest generation of multiple-resource attention models provide promising capabilities of broad applicability in real technical applications.

Aiming at estimating attention levels using pervasive computing technologies, we depend on overt, observable somatic indicators of human attention which can be measured, quantified and interpreted. Concerning the individual, the analysis of Visual Attention represents the most intuitive and most frequently pursued approach [13], focusing mainly on stimulus-driven (bottom-up) attention mechanisms in contrast to expectation-driven (top-down) aspects of attention control. Modeling of Visual Attention is widely based on visual saliency as characteristics describing attention capturing capabilities of input stimuli. There are numerous models mimicking biological findings to best possible reproduce human eye movement patterns as, e.g., cognitive models [14][15], Bayesian models [16–18], decision theoretic models [19], information theoretic models [20][21] or alternatively pure data-driven models [22].

In technical applications, the so-called *Visual Focus of Attention* (VFOA) has become the main representation of the estimated orientation of attention, covering analysis of gaze direction [23], eye movements [24, 25], saccades and fixations [26, 27] or head orientation depending on sensor data quality. In addition to visual attention, overt attentional behavior is interpreted in public area scenarios, describing competitive

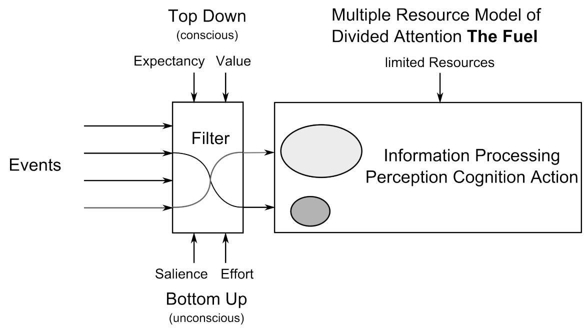


Figure 1. SEEV-Attention Model by Wickens [11] in which perception filters are controlled via Salience, Expectation, Effort and Value.

selection and switched attention processes. Smith et al. [28] created a head tracking in an out-of-home advertisement scenario in which subjects were evaluated as *focused* if their gaze was directed at the display longer than a certain threshold. Leykin and Hammoud [29] used single camera surveillance footage to estimate the area and direction of interest of pedestrians with head pose and movement direction as features. Ozturk et al. [30] investigated the orientation of people in indoor environments, interpreting orientation as indicator for visual focus of attention. Yakiyama et al. [31, 32] estimated attention levels towards target objects on the basis of computed distance, orientation and movement speed. Quek et al. [33] analyzed feet positioning and movement orientation to estimate the the orientation of perception.

In this paper, we will explore and validate physical effort as a crucial and expressive indicator of attentional behavior in the context of public display scenarios. For this purpose, we employ predictions of attentional behavior derived from Wickens' SEEV attention model, which are based on a previously published, general effort-based behavior description [34]. We analyze behavior to calculate physical effort scores that need to be invested for the potential perception of displayed information to identify significant correlations between physical effort, behavior and resulting engagement and attention. The identified correlations are supposed to validate physical effort as a reliable indicator for human attention estimation in public display scenarios.

II. Exploration of the SEEV attention model

In his elaborate research, Wickens created an extensive attention model [11], [35], which describes attention as an information processing system with multiple channels and resource types, which is equipped with a filtering unit driven by conscious (top-down) and unconscious (bottom-up) processes (cf. Figure 1). The central message of this approach is, that parallel cognitive processes are possible without interference as long as they differ in occupied channels and required types of resources (e.g., visual tasks can be executed in parallel to auditory tasks).

In its computational annotation, the so-called SEEV-attention model describes the probability to attend an area of interest via a linear equation, which is based on four components and their respective scaling factors - namely, the contributive factors of stimulus attractiveness (Saliency **S**), expected information entropy (Expectancy **E**x) and importance of the accustomed information (Value **V**) in contrast to the physical and mental Effort **E**f required for the assessment of the respective information (cf. Equ. 1):

$$P(A) = s \cdot \mathbf{S} - ef \cdot \mathbf{E}f + (ex \cdot \mathbf{E}x + v \cdot \mathbf{V}) \qquad (1)$$

Due to its linear and intuitive design, the SEEV model has successfully found application in several different scenarios as (i) aviation: modeling of attentional behavior and attention errors of pilots and air traffic controllers [36–38] (ii) automotive sector: simulation of car driver behavior [39–42] as well as information and infotainment systems in the automotive environment and mobile service interfaces [43] (iii) general computer interaction research [44] as well as (iv) perception and reaction to alerts and notifications [45]. The listed approaches use eye tracking or other sensorial data for behavior input and compare predicted and observed behavior to evaluate the applicability of the SEEV model and the respective author's developments, whereas, depending on the available data and given use-case, the SEEV-equation were simplified or generalizing assumptions were made.

A. *The SEEV model in a public display scenario*

The control parameters of the SEEV model Figure 1) require some more elaboration and interpretation regarding our specific use-case of pervasive display scenarios:

- **Value** represents the personal evaluation of the relevance of events, stimuli or perceived information. It is based on personal experiences and preferences, thus together with expectancy describes the individual intrinsic components of the SEEV attention model.
  As these intrinsic parameters are not accessible in the given experiment (anonymous users, no preferences or previous behavior history available), this parameter is deprived from our knowledge and control. As the presented use-case at an annual sports fair represents an event with voluntary presence, we are confident to assume positive overall associations of the audience towards the event in general and to simplify personal value to a constant positive average score.
- **Expectancy** describes the expected information bandwidth or entropy of an object, event or area of interest. Expectancy is closely related to personal experience of information consumption and especially thus linked to Value. Areas of interest that provided useful, entertaining or helpful information in the past are more likely to be attended and frequented in the future [46].
  Large-scale displays have become a fundamental part of public life and represent established omnipresent and continuous sources of information. They have evolved towards an essential role in the public domain with an associated high bandwidth of information, regardless of the associated personal relevance of displayed information. Again, this gives us the confidence to reduce model complexity by setting the expectancy score to a constant positive value.
- **Effort** represents the physical or mental effort, necessary to change current behavior to attend areas of interest and assess the available information. For example, this includes overcoming physical distances as well as exhausting cognitive filtering processes in noisy environments. Effort represents the only inhibitor of attention allocation in the SEEV attention model.

In our preliminary work [34], [47] we used invested physical effort to interpret engagement and attention to public displays. In this work, we try to validate these findings by employing the SEEV model in which effort - as the only inhibitory factor - plays a crucial role in the assessment of information. Hence, the required effort for content perception will serve as the main variable to be observed and evaluated.

- **Saliency** expresses the attractiveness of an event or stimulus mainly regarding bottom up processes, often affected significantly by contrast. Salient events differ from their environment via a single or several perceptional characteristics (e.g., color, size, movement, volume, frequency range, etc.). An overload with irrelevant stimuli is reported to reduce performance [48] and highly salient stimuli are likely to capture attention even when irrelevant to the task [49].
  Besides Effort, Saliency represents the second crucial factor in this setting. In our approach, we try to assess to which extent effort alone is sufficient to describe observed attention behavior. Due to that goal, we employed highly salient video content displaying extreme sports performances to ensure high and constant attractiveness. As a consequence, we again assume Saliency as a constant score in favor of a direct sensitivity to effort as the variable of interest, in the awareness of the over-simplification of the model and the need to take this reduction of complexity into consideration in the discussion and evaluation process.

The consequences of the described assumptions and simplifications are presented in the adapted equation for SEEV computation:

$$\begin{aligned} P_A(t) &= (s \cdot \mathbf{S} + ex \cdot \mathbf{E}x + v \cdot \mathbf{V}) - ef \cdot \mathbf{E}f(t) && (2)\\ &= (C) - ef \cdot \mathbf{E}f(t) && (3)\\ &= 1 - ef \cdot \mathbf{E}f(t) && (4) \end{aligned}$$

Value, Expectancy and Saliency are combined to a single constant value and in the context of a description of probability, to the maximum probability of 1. This results in an indirect proportional relation between attentional behavior and required effort, which will be experimentally verified in the following. The exact value of C is not relevant in the following evaluation via correlation since constant values do not have any influence on correlation scores. Since we only want to investigate the dependency of attention from effort behavior, it is an intuitive step to set C to 1 as it decreases potential misinterpretations.

III. Experiment description

To find significant correlations between attention distributions predicted by SEEV probabilities and observed and annotated attentional behavior, we employ experimental data, which already served as input for preliminary works in [47]. The experiment was set up at a public sports event in the scope of the Vienna City Marathon. A large-scale public display integrated in a standard City Light cabinet (1,86 x 1,30 m) equipped with two 50” monitors and a depth camera sensor was installed in the entrance area of the event, with the sensor covering the area in front of the display (cf. Figure 2) with an horizontal opening angle of 57 ° and a maximal depth range of 6 m. The display itself did not show any interactive behavior, but employed depth sensors were only used for data collection.

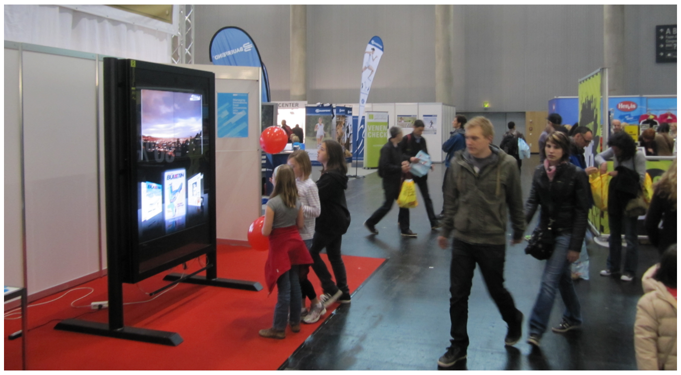


Figure 2. Setup of large-scale public display at Vienna Sports World Fair. Depth cameras were used to detect and extract behavior features for effort computation.

The recorded depth data was afterwards used to apply OpenNI [50] and Primesense [51] skeleton tracking algorithms and extract skeleton joint data, which was used to compute and analyze behavior [34].

IV. Behavior Analysis and Creation of Data set

A. Computation of Minimum Required Effort - MRE

In earlier works, we interpreted the overall amount of invested effort to deduce the quality of engagement with a public display (Directed Effort [34]). A person in a public space has the following independent degrees of freedom regarding general movement in the horizontal plane (i) movement speed v (ii) movement direction φ (iii) body orientation. Similar to earlier computations of effort we stick to the strict additive separation of these behavioral dimensions and define thresholds for possible perception for each single parameter. We employ the following annotation in the ongoing: $i \in v, \sigma, \varphi, \tau$ as placeholder for the respective behavioral dimensions, as well as i_T as representation of the respective perceptional threshold.

As the analysis of the SEEV model is not directed at the quality but at the probability of attention, we are now aiming for the effort threshold that needs to be invested to enable the perception of the displayed content (MRE). Hence, some adaptations to the computation of invested effort had to be made in comparison to earlier processes: Body orientation is split up into head orientation σ and upper body orientation τ to better model head turns in relation to the overall body orientation.

The computation of all parameters is based on the ratio of the difference between current behavior and the respective thresholds i_T to the overall possible change in this specific parameter dimension (cf. Equ. 5-9). This provides a percental rating to what degree the person needs to change its activities in the respective behavioral dimension to enable the perception of displayed content. If possible, the selection of these thresholds was based on scientific research, otherwise they have been set according to observations and the personal experience of the authors. The respective maximum reference scores have been set via empiric analysis of the overall data set.

B. Speed

Movement speed is characterized as the magnitude of the movement vector $v(t) = |\overrightarrow{v(t)}|$. The speed threshold has been set to a score, which resembles a medium walking pace in the authors implementations. In figure 3(a), this is visualized via a

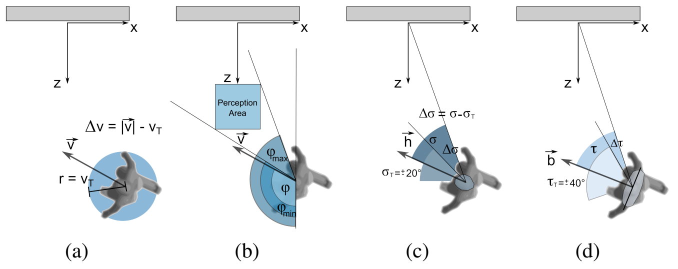


Figure 3. Computation of Minimum Required Effort in the dimensions of (a) movement speed $v(t)$, (b) movement direction $\varphi(t)$, (c) head orientation $\sigma(t)$ and (d) upper body orientation $\tau(t)$.

centric circle with radius v_T around the person center of mass location.

C. Movement Direction

According to the Focus-Nimbus Aspects Attributes and Resources (FN-AAR) model [52], which is based the Focus/Nimbus (FN) awareness model, there are areas of comfortable perception of displayed information. We employed the area as visualized in figure 3(b) as reference location for the perception threshold and to evaluate movement direction to overcome this hindrance. The position and movement vector of each proband is analyzed to compute whether the person is moving towards this perception zone, and if not, the required effort for behavior change (movement angle) is computed.

$$E(t) = e_v \cdot E_v(t) + e_\varphi \cdot E_\varphi(t) + e_\sigma \cdot E_\sigma(t) + e_\tau \cdot E_\tau(t) \tag{5}$$

$$E_v(t) = \begin{cases} \frac{v(t)-v_T}{\Delta v_{max}}, & \text{if } v(t) > v_T \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

$$E_\varphi(t) = \begin{cases} \frac{\varphi(t)-\varphi_T}{\Delta \varphi_{max}}, & \text{if } \varphi_{max} > \varphi(t) > \varphi_{min} \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

$$E_\sigma(t) = \begin{cases} \frac{\sigma(t)-\sigma_T}{\Delta \sigma_{max}}, & \text{if } \sigma(t) > \sigma_T \\ 0, & \text{otherwise} \end{cases} \tag{8}$$

$$E_\tau(t) = \begin{cases} \frac{\tau(t)-\tau_T}{\Delta \tau_{max}}, & \text{if } \tau(t) > \tau_T \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

D. Head Orientation

Head orientation is interpreted as general Visual Focus of Attention as a low-level feature of eye gaze and description of general field of vision. The captured depth data did not provide sufficient resolution for actual eye tracking algorithms over the distance of several meters, so we were restricted to head orientation measurements derived from depth data. For this purpose, we employed the implementations by Fanelli et al. [53]. The threshold σ_t for head orientation was set to 20°, as this angle between current eye position and goal of the saccade has been reported by Kahneman [8] as the limit from which head movements replaces eye movements for the assessment of the information (cf. Figure 3(c)). Below 20° head orientation in relation to the display location, we assume a perception of the display content as sufficiently probable.

E. Upper Body Orientation

The threshold for required turn of the upper body has been set to 40° (cf. Figure 3(d)). This seemingly arbitrarily defined score is the result of empiric observations and personal experiences and may be necessary to be replaced by more scientifically founded studies.

F. Ground Truth Labeling

Due to the lack of qualitative attention metrics, the evaluation of attention models represents a challenging issue. The main challenge in this context is obtaining an objective ground truth labeling of observed behavior. Subjective reports from subjects provide helpful assistance, yet in our case could not provide the required temporal resolution and detail.

In this work, we apply manually transcriptions that have already been applied in earlier works, which are combined from (i) real-time observations and annotations that were obtained in the gathering of the experiment data, (ii) detailed frame-wise post-hoc labeling of behavior scores and (iii) interview results from subjects regarding their overall awareness of the display and perceived content. The hand-labeled behavior scores are based on detailed categories of behavior that are associated to a numeric scale (0-10) and associated to 5 more general classes of behavior, representing increasing behavior change and engagement with the pervasive display.

In the following, we employ the general five behavioral classes of (0) No perception - display of of field of view, (1) Selective Perception - display in peripheral visual range, (2) Switched Attention - display actively in wandering gaze area, (3) Focused Attention - conscious perception, distinct behavior change (4) Sustained Attention - lasting perception, fundamental behavior change. Please refer to [47] for the detailed behavior assignment for the 11-tier scale and 5-class behavior segmentation.

G. Overall dataset

The resulting dataset is based on 188 probands that were observed and analyzed, resulting in an overall of 27'891 frames, which hold required effort for the respective behavioral dimension, prediction of attention probability and labeled ground truth of observed attentional behavior.

TABLE I. CORRELATION SCORES OF SINGLE EFFORT DIMENSIONS

Effort Parameter	Correlation Score
E_v	0.7138437
E_φ	0.3944934
E_σ	0.4232690
E_τ	0.2348277

V. Results and Discussion

As a first step, we tried to identify the weighting parameters e_i (cf. Equ. 5) to find an optimal contribution of the single behavioral dimensions for the aspired modeling of actual attention behavior. For this purpose, we computed the correlations between the single effort components and the hand-labeled ground truth data to evaluate the contributivity of the single parameters. The single correlation scores, averaged over the complete dataset, are displayed in Table I.

As can be observed, movement speed shows by far the most promising correlation of the four parameters. The minor

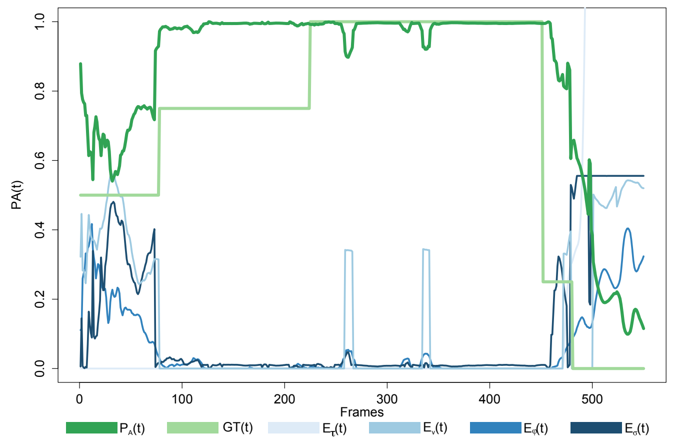


Figure 4. Exemplary sequence of behavioral dimensions E_v, E_φ, E_σ, E_τ, overall predicted attention probability $P_A(t) = 1 - E(t)$ and ground truth labels for a single proband over time (single correlation score: 0.84).

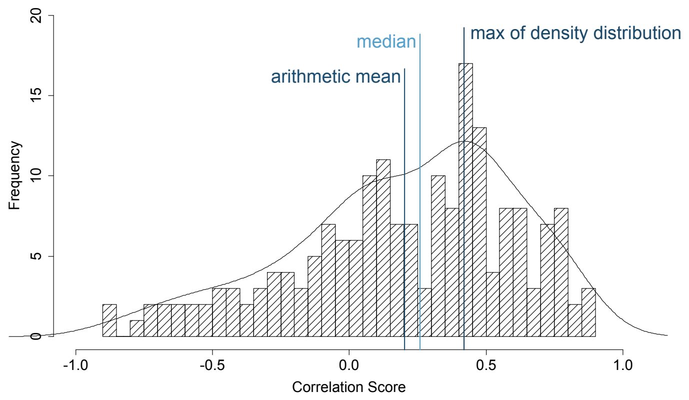


Figure 5. Histogram and density function (Gaussian kernel) of correlation results per person, with different statistical interpretative norms (mean, median, max. of density function).

performance of other parameters especially head and body orientation is partly related to the varying quality of skeleton tracking data. Yet these other behavioral aspects may still push the overall correlation between model prediction and observation scores. For this purpose, we used the correlation scores as weighting parameters:

$$e_i = \frac{cor(E_i)}{\sum_i cor(E_i)} \tag{10}$$

A. Evaluation

For the analysis of the collected dataset, we employed the implementation of Pearson's product-moment correlation in R [54]. An exemplary plot of the sequence of computed scores for a sample of the dataset is displayed in Figure 4. It shows how the single computed parameters add up to the overall Effort score and relate to annotated behavior.

The overall evaluation of the gathered dataset using correlative dependencies requires some discussion. Generally, the computation of an adequate overall correlation score from different single correlation results is not trivial, as the database items differ in sample sizes and significance scores. The sample sizes range from 14 to 2321 captured frames per person with an average of 148.35 frames. This imbalance between data samples represents the main challenge regarding the computation of an expressive overall correlation score.

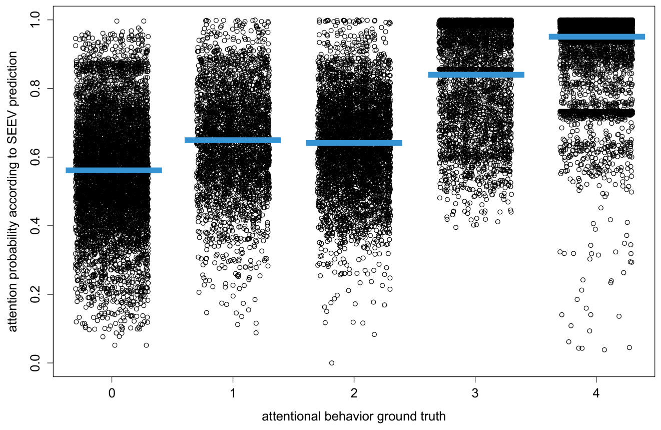


Figure 6. Scatter plot of computed attention probability scores for complete dataset (27'891) frames in relation to associated attention behavior ground truth. Horizontal bars indicate the arithmetic mean. The overall distributions show strong proportional tendencies.

To approach this issue of evaluation, we pursue two different evaluation approaches, neither of which can claim to give an absolute overall correlation score, but which will show the range of correlation dependencies and give an impression on the overall performance of our approach.

1) Averaging over single probands: Simple averaging over single subject correlation scores as overall correlation result based will show bias towards shorter sample sizes. As longer samples show a better correlation score, this process will underestimate the actual system performance. Furthermore, it is debatable, which averaging mechanism best represents correlation results. The histogram of obtained single correlation scores is visualized in Figure 5. In this case, the arithmetic mean seems to be too sensitive to far outliers, whereas the median (0.263) and the computed density function (0.413), employing a Gaussian kernel seem to better represent the overall distribution of results.

2) Creation of an overall Correlation: We can avoid the problem of averaging of incomparable samples via conflating all computed single temporal $P_A(t)$ and GT curves into a single, encompassing dataset. This would be equivalent to a single user being present in the scene for continuous 27'891 frames or 15h and 29min. The result is computed as the overall correlation between computed (effort-based) attention probabilities and hand-labeled observations, which computes to an overall correlation score of 0.749. This interpretation of correlation scores provides a far better rating of the dependencies between behavior and attention distribution, yet is probably prone to an overestimation since longer samples will cause heavier weights and again a imbalanced evaluation.

In the scatter plot in Figure 6, the general relation between predicted attention probabilities and overt attention behavior can be observed. The mean scores and general distributions show the expected strong proportional tendency, which is expected to increase under consideration of not only physical effort, but stimulus saliency as driving forces of attentional behavior.

Yet, this approach provides a new issue in the evaluation of these results in the assessment of significance. Usually, significance scores of $p < 0.05$ or $p < 0.02$ are claimed for reliable dependencies where coincidences are excluded. In the given evaluation, the problem arises from the extensive size of the dataset. As the significance score p is directly related to the size of the dataset it will turn against 0 ($2.2 \cdot e^{-16}$ in our case) for large data sets, a problem, which is already established in the scientific community. Thus the significance score loses its validity as an indicator in our case. Lin et al. [55] proposed to use confidence intervals instead of p as a more expressive evaluator regarding the validity of correlation results. In our case, the 95% confidence interval ranges from 0.743 to 0.754, a range, which makes us confident to report a significant relationship between required physical effort and attention behavior.

VI. Conclusion & Outlook

In this paper, we have identified and demonstrated significant correlations between attention behavior predicted by the established SEEV computational attention model, based on minimum required physical effort as descriptor, and actual observed attentional behavior in an empiric experimental study. In a critical discussion of different approaches towards a general evaluation of an absolute correlation score, we estimate the true correlation score between the overestimated score of 0.749 for overall correlation computation and the underestimated score of 0.413 for per person correlation averaging.

Taking into account, that the major aspect of time-dependent saliency has been ignored in the modeling and evaluation process, the obtained results absolutely show promising potentials regarding physical effort as a suitable attention modeling parameter in public domains. A future inclusion of real-time saliency evaluation of displayed visual and audio content is expected to further boost our attention modeling process. Additionally, investigating surplus effort, exceeding the necessary thresholds for perception might represent a suitable indicator for the quality of engagement and attention. Finally, in spite of undeniable efforts, the statistical evaluation requires further research regarding the computation of a suitable, explicit overall correlation score.

References

[1] T. H. Davenport and J. C. Beck, "The attention economy," Ubiquity, vol. 2001, no. May, May 2001. [Online]. Available: http://doi.acm.org/10.1145/376625.376626

[2] W. James, The Principles of Psychology, Vol. 1. Dover Publications, Jun. 1950. [Online]. Available: http://www.worldcat.org/isbn/0486203816

[3] D. E. Broadbent, Perception and communication, 3rd ed. Oxford: Pergamon Press, 1969.

[4] K. J. Craik, "Theory of the human operator in control systems1," British Journal of Psychology. General Section, vol. 38, no. 2, 1947, pp. 56–61.

[5] A. M. Treisman and G. Gelade, "A feature-integration theory of attention," Cognitive Psychology, vol. 12, no. 1, 1980, pp. 97–136.

[6] J. A. Deutsch and D. Deutsch, "Attention: Some theoretical considerations," Psychological review, vol. 70, 1963, pp. 80–90.

[7] N. Lavie, A. Hirst, J. W. d. Fockert, and E. Viding, "Load Theory of Selective Attention and Cognitive Control," Journal of Experimental Psychology: General, vol. 133, no. 3, 2004, pp. 339–354.

[8] D. Kahneman, Attention and effort. Englewood Cliffs, N.J.: Prentice-Hall, 1973.

[9] H. P. Bahrick and C. Shelly, "Time sharing as an index of automatization." Journal of Experimental Psychology, vol. 56, no. 3, 1958, p. 288.

[10] W. Schneider and R. M. Shiffrin, "Controlled and automatic human information processing: I. Detection, search, and attention," Psychological Review, vol. 84, no. 1, 1977, pp. 1–66.

[11] C. D. Wickens, "Multiple resources and mental workload." Human Factors, vol. 50, no. 3, 2008, pp. 449–455. [Online]. Available: http://dblp.uni-trier.de/db/journals/hf/hf50.htmlWickens08a

[12] E. I. Knudsen, "Fundamental components of attention." Annual Review of Neuroscience, vol. 30, no. 1, 2007, pp. 57–78. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/17417935

[13] A. Borji and L. Itti, "State-of-the-art in visual attention modeling," Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 35, no. 1, 2013, pp. 185–207.

[14] L. Itti, C. Koch, and E. Niebur, "A model of saliency-based visual attention for rapid scene analysis," IEEE Transactions on pattern analysis and machine intelligence, vol. 20, no. 11, 1998, pp. 1254–1259.

[15] O. Le Meur, P. Le Callet, D. Barba, and D. Thoreau, "A coherent computational approach to model bottom-up visual attention," Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 28, no. 5, 2006, pp. 802–817.

[16] A. Oliva, A. Torralba, M. S. Castelhano, and J. M. Henderson, "Top-down control of visual attention in object detection," in Image Processing, 2003. ICIP 2003. Proceedings. 2003 International Conference on, vol. 1. IEEE, 2003, pp. I–253.

[17] L. Zhang, M. H. Tong, T. K. Marks, H. Shan, and G. W. Cottrell, "Sun: A bayesian framework for saliency using natural statistics," Journal of vision, vol. 8, no. 7, 2008, p. 32.

[18] J. Li, Y. Tian, T. Huang, and W. Gao, "Probabilistic multi-task learning for visual saliency estimation in video," International Journal of Computer Vision, vol. 90, no. 2, 2010, pp. 150–165.

[19] D. Gao and N. Vasconcelos, "Discriminant saliency for visual recognition from cluttered scenes," in Advances in neural information processing systems, 2004, pp. 481–488.

[20] R. Rosenholtz, "A simple saliency model predicts a number of motion popout phenomena," Vision research, vol. 39, no. 19, 1999, pp. 3157–3163.

[21] N. Bruce and J. Tsotsos, "Saliency based on information maximization," in Advances in neural information processing systems, 2005, pp. 155–162.

[22] X. Hou and L. Zhang, "Saliency detection: A spectral residual approach," in Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on. IEEE, 2007, pp. 1–8.

[23] R. Stiefelhagen, M. Finke, J. Yang, and A. Waibel, "From gaze to focus of attention," in Visual Information and Information Systems. Springer, 1999, pp. 765–772.

[24] E. Kowler, "Eye movements: The past 25years," Vision research, vol. 51, no. 13, 2011, pp. 1457–1483.

[25] M. Hayhoe and D. Ballard, "Eye movements in natural behavior," Trends in cognitive sciences, vol. 9, no. 4, 2005, pp. 188–194.

[26] S. P. Liversedge and J. M. Findlay, "Saccadic eye movements and cognition," Trends in cognitive sciences, vol. 4, no. 1, 2000, pp. 6–14.
[27] J. E. Hoffman and B. Subramaniam, "The role of visual attention in saccadic eye movements," Perception & psychophysics, vol. 57, no. 6, 1995, pp. 787–795.
[28] K. C. Smith, S. O. Ba, J.-M. Odobez, and D. Gatica-Perez, "Tracking attention for multiple people: Wandering visual focus of attention estimation," IDIAP, Tech. Rep., 2006.
[29] A. Leykin and R. Hammoud, "Real-time estimation of human attention field in lwir and color surveillance videos," in Computer Vision and Pattern Recognition Workshops, 2008. CVPRW '08. IEEE Computer Society Conference on, june 2008, pp. 1 –6.
[30] O. Ozturk, T. Yamasaki, and K. Aizawa, "Tracking of humans and estimation of body/head orientation from top-view single camera for visual focus of attention analysis," in Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on, 27 2009-oct. 4 2009, pp. 1020 –1027.
[31] Y. Yakiyama, N. Thepvilojanapong, M. Iwai, O. Mihirogi, K. Umeda, and Y. Tobe, "Observing real-world attention by a laser scanner," IPSJ Online Transactions, vol. 2, 2009, pp. 93–106.
[32] N. Thepvilojanapong, Y. Yakiyama, O. Mihirogi, M. Iwai, K. Umeda, Y. Tobe, and R. Shibasaki, "Evaluating people's attention in the real world," in ICCAS-SICE, 2009, aug. 2009, pp. 3702 –3709.
[33] F. Quek, R. Ehrich, and T. Lockhart, "As go the feet...: on the estimation of attentional focus from stance," in Proceedings of the 10th international conference on Multimodal interfaces, ser. ICMI '08. New York, NY, USA: ACM, 2008, pp. 97–104. [Online]. Available: http://doi.acm.org/10.1145/1452392.1452412
[34] B. Gollan and A. Ferscha, "A generic effort-based behavior description for user engagement analysis," in Physiological Computing Systems. Springer, 2014, pp. 157–169.
[35] C. D. Wickens, "Multiple resources and mental workload," Human Factors: The Journal of the Human Factors and Ergonomics Society, vol. 50, no. 3, 2008, pp. 449–455.
[36] C. Wickens, J. McCarley, and K. Steelman-Allen, "Nt-seev: A model of attention capture and noticing on the flight deck," Proceedings of the Human Factors and Ergonomics Society Annual Meeting, vol. 53, no. 12, 2009, pp. 769–773. [Online]. Available: http://pro.sagepub.com/content/53/12/769.abstract
[37] C. D. Wickens, J. S. McCarley, A. L. Alexander, L. C. Thomas, M. Ambinder, and S. Zheng, "Attention-situation awareness (a-sa) model of pilot error," Human performance modeling in aviation, 2008, pp. 213–239.
[38] X. Wu, X. Wanyan, and D. Zhuang, "Attention allocation modeling under multifactor condition," Journal of Beijing University of Aeronautics and Astronautics, vol. 8, no. 39, 2013, p. 1086.
[39] W. J. Horrey, C. D. Wickens, and K. P. Consalus, "Modeling drivers' visual attention allocation while interacting with in-vehicle technologies." Journal of Experimental Psychology: Applied, vol. 12, no. 2, 2006, p. 67.
[40] J.-T. Wong and S.-H. Huang, "A microscopic driver attention allocation model," Advances in Transportation Studies an International Journal, 2011, pp. 53–64.
[41] R. Kaul, M. Baumann, and B. Wortelen, "The influence of predictability and frequency of events on the gaze behaviour while driving," in Human Modelling in Assisted Transportation. Springer, 2011, pp. 283–290.
[42] N. D. Cassavaugh, A. Bos, C. McDonald, P. Gunaratne, and R. W. Backs, "Assessment of the seev model to predict attention allocation at intersections during simulated driving," in 7th International Driving Symposium on Human Factors in Driver Assessment, Training, and Vehicle Design, no. 52, 2013.
[43] J. Niemann, V. Presse, J. Reissland, and A. Naumann, "Developing a user-centered mobile service interface based on a cognitive model of attention allocation," in Human-Computer Interaction. Springer, 2010, pp. 50–57.
[44] B. Gore, B. Hooey, C. Wickens, and S. Scott-Nash, "A computational implementation of a human attention guiding mechanism in midas v5," in Digital Human Modeling, ser. Lecture Notes in Computer Science, V. Duffy, Ed. Springer Berlin Heidelberg, 2009, vol. 5620, pp. 237–246. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-02809-0_26
[45] J. B. Mulligan, "Factors influencing scanning for alerts," Journal of Vision, vol. 10, no. 15, 2010, pp. 69–69.
[46] J. Müller, D. Wilmsmann, J. Exeler, M. Buzeck, A. Schmidt, T. Jay, and A. Krüger, "Display blindness: The effect of expectations on attention towards digital signage," in Pervasive Computing. Springer, 2009, pp. 1–8.
[47] B. Gollan, B. Wally, and A. Ferscha, "Automatic human attention estimation in an interactive system based on behaviour analysis," Proc. EPIA 2011, 2011.
[48] W. Bacon and H. Egeth, "Overriding stimulus-driven attentional capture," Perception & Psychophysics, vol. 55, no. 5, 1994, pp. 485–496. [Online]. Available: http://dx.doi.org/10.3758/BF03205306
[49] S. Yantis, "Stimulus-driven attentional capture," Current Directions in Psychological Science, 1993, pp. 156–161.
[50] OpenNI User Guide, OpenNI organization, November 2010, last viewed 19-01-2011 11:32. [Online]. Available: http://www.openni.org/documentation
[51] Prime Sensor NITE 1.3 Algorithms notes, PrimeSense Inc., 2010, last viewed 19-01-2011 15:34. [Online]. Available: http://www.primesense.com
[52] S. Benford, A. Bullock, N. Cook, P. Harvey, R. Ingram, and O.-K. Lee, "From rooms to cyberspace: models of interaction in large virtual computer spaces," Interacting with Computers, vol. 5, no. 2, 1993, pp. 217–237.
[53] G. Fanelli, T. Weise, J. Gall, and L. Van Gool, "Real time head pose estimation from consumer depth cameras," in Pattern Recognition. Springer, 2011, pp. 101–110.
[54] R Development Core Team, R: A Language and Environment for Statistical Computing, R Foundation for Statistical Computing, Vienna, Austria, 2012, ISBN 3-900051-07-0. [Online]. Available: http://www.R-project.org/
[55] M. Lin, H. C. Lucas Jr, and G. Shmueli, "Research commentary-too big to fail: large samples and the p-value problem," Information Systems Research, vol. 24, no. 4, 2013, pp. 906–917.

Unconventional Computing Using Evolution-in-Nanomaterio: Neural Networks meet Nanoparticle Networks

Klaus Greff[†], Ruud van Damme[*], Jan Koutník[†], Hajo Broersma[*], Julia Mikhal[*],
Celestine Lawrence[*], Wilfred van der Wiel[*], and Jürgen Schmidhuber[†]
[†]IDSIA, USI-SUPSI, Galleria 2, 6928 Manno-Lugano, Switzerland
Email: {klaus,hkou,juergen}@idsia.ch
[*]Faculty of EEMCS, CTIT and MESA+ Institutes for ICT Research and Nanotechnology
University of Twente, The Netherlands
Email: {r.m.j.vandamme,h.j.broersma,c.p.lawrence,w.g.vanderwiel}@utwente.nl

***Abstract*—Recently published experimental work on evolution-in-materio applied to nanoscale materials shows promising results for future reconfigurable devices. These experiments were performed on disordered nano-particle networks that have no predefined design. The material has been treated as a black-box, and genetic algorithms have been used to find appropriate configuration voltages to enable the target functionality. In order to support future experiments, we developed simulation tools for predicting candidate functionalities. One of these tools is based on a physical model, but the one we introduce in this paper is based on an artificial neural network. The advantage of this newly presented approach is that, after training the neural network to match either the real material or its physical model, it can be configured using gradient descent instead of a black-box optimisation. The experiments we report here demonstrate that the neural network can model the simulated nano-material quite accurately. The differentiable, neural network-based material model is then used to find logic gates, as a proof of principle. This shows that the new approach has great potential for partly replacing costly and time-consuming experiments with the real materials. Therefore, this approach has a high relevance for future computing, either as an alternative to digital computing or as an alternative way of producing multi-functional reconfigurable devices.**

***Keywords*–nanoparticle network; neural network; simulation; unconventional computation; evolution-in-nanomaterio.**

I. Introduction and Motivation

Within the EU-project NASCENCE [4], disordered networks of gold nano-particles have been successfully used to produce reconfigurable logic with a very high degree of stability and reproducibility [3]. This breakthrough presents a proof of principle that indicates very promising prospects for using this material to perform more complicated computational tasks. In order to predict candidate computational tasks, but avoiding the waste of scarce and expensive resources involved in experimentally exploring these nano-particle networks, we developed simulation tools for examining the capabilities of these material systems. One of them [7] is an extension of existing tools for simulating nano-particle interactions, like SPICE [11] or SIMON [18]. The latter tools are all based on Monte-Carlo simulations, using a physical model, and have been validated for designed systems with small numbers of particles. Although these methods can, in principle, handle arbitrary systems of any size, their scalability is a serious issue. Moreover, nano-particle networks like the ones used in [3] to date cannot be produced according to a predefined specific design. Therefore, due to their disordered nature, it is not possible to use an accurate physical model for such material systems.

In this paper, an alternative approach is introduced. This novel approach is based on training artificial Neural Networks in order to model and investigate the nano-particle networks. Neural Networks (NN; [10] [13] [19]) have proven to be powerful function approximators and have, especially recently, been applied to a wide variety of domains with great success [5] [6] [14]. Being essentially treated as black-boxes themselves, NNs do not facilitate a better understanding of the underlying quantum-mechanical processes that take place in the material. For that purpose, physical models are more appropriate. But, in contrast to physical models, NNs provide differentiable models and thus offer interesting possibilities to explore the computational capabilities of the nano-material. In the sequel, NNs will be used, in particular, to search for configurations of input voltages such that the material computes different Boolean logic functions, such as AND, OR, NOR, NAND, and XOR.

To enable the exploration of the computational capabilities of the material by an NN, the NN needs to be trained first with data collected from the measurements on the material. Since a physical model and an associated validated simulation tool for the nano-particle networks have already been developed [7], such training data can be obtained from the simulated material. This also provides an opportunity for predicting functionalities in small nano-particle networks that have not been fabricated yet. This in turn can inform electrical engineers on the minimum requirements necessary for obtaining such functionalities, without the burden of costly and time-consuming fabrication and experimentation. As soon as the NN has been trained, searching for arbitrary target functions is very fast, and can happen without any access to the material or its physical model.

The rest of the paper is organised as follows. Section II provides some technical details on the gold nano-particle networks that have been used in the experimental work [3]. This section also describes the choices that have been made and that form the basis for the physical-model based simulation tool of [7], combining a genetic algorithm with Monte-Carlo simulations for charge transport. Section III presents simulation results obtained with these tools for an example network. Section IV shows how an NN was trained for modelling the example network, using data collected from the physical-model based simulation tool. An analysis of the results is presented in Section V, followed by a short section with conclusions as

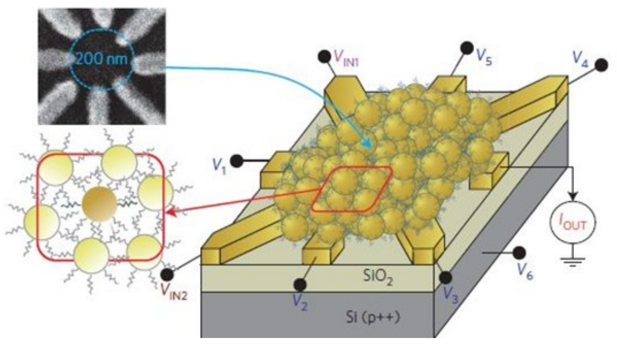


Figure 1. Illustration of a disordered network of gold NPs

well as an outlook to future work.

II. Nano-particle Networks and their Simulation

In collaboration with the NanoElectronics group at the MESA+ institute of the University of Twente, networks consisting of commercially available nano-particles of size 5-20 nm consisting of gold, and junctions of alkanedithiol of length 1-3 nm have been produced. The alkanedithiols stick to the metal and can form junctions (tunnel barriers) between particles. Figure 1 shows an illustration of such a network. The central circular region is about 200 nm in diameter. More details on the production process can be found in [3].

The networks investigated there are relatively large (in the order of a hundred particles) and disordered. The transport of electrons is governed by the Coulomb blockade effect [8]: transport is blocked, except at almost discrete energy levels; there exactly one electron can jump. The dynamics of such a system is governed by stochastic processes: electrons on particles can tunnel through junctions with a certain probability. For such systems, there are basically two simulation methods to one's disposal: Monte-Carlo Methods and the Master Equation Method [16] [17]. Since the number of particles is large, the Monte-Carlo Method is the best candidate. This method simulates the tunnelling times of electrons stochastically. To get meaningful results, one needs to run the algorithm in the order of a million times. Doing so, the stochastic process gives averaged values of the charges, currents, voltages, etc. More details on the simulation tool can be found in [7].

III. An Illustrative Example

As an example which is still relatively small and manageable, but shows interesting features, the described methods have been explored on the symmetric 4×4-grid consisting of the components shown in Figure 2.

In Figure 2, the 16 green dots represent the nano-particles; in between are the tunnelling junctions, with fixed C and R values. The two input leads and the single output lead are depicted as I_1 and I_2 and O, respectively. Voltages are applied to the configuration leads V_1-V_9, according to a Genetic Algorithm, and also to a back gate; this back gate is connected through tunnel barriers (a silicon oxide layer) to all nano-particles (for convenience we have not shown the back gate in the figure).

The fitness of the sets of configuration voltages is determined by how close the output for the four input combinations

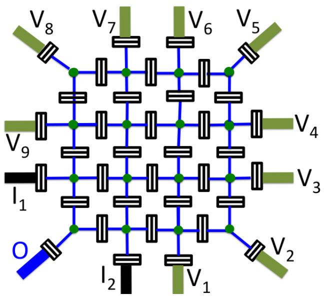


Figure 2. A symmetric 4×4-grid of 16 nano-particles with leads

of the Boolean truth table is to the desired logic. We omit the details due to page restrictions. Details can be found in [7].

1) Evolved Boolean Logic: Applying the developed simulation tool to the small network of Figure 2, it was possible to evolve all basic Boolean logic gates, using different computed (simulated and optimised) settings of the values of the free variables (the configuration leads voltages and the back gate voltage). The solutions for four of the cases, namely AND, NAND, OR and XOR, are illustrated as contour plots in Figure 3. The four plots are functions of the two input signals; the voltages of both inputs range from 0 to 10 mV, horizontally as well as vertically; the colour scheme ranges from blue for small values to red for high values of the output.

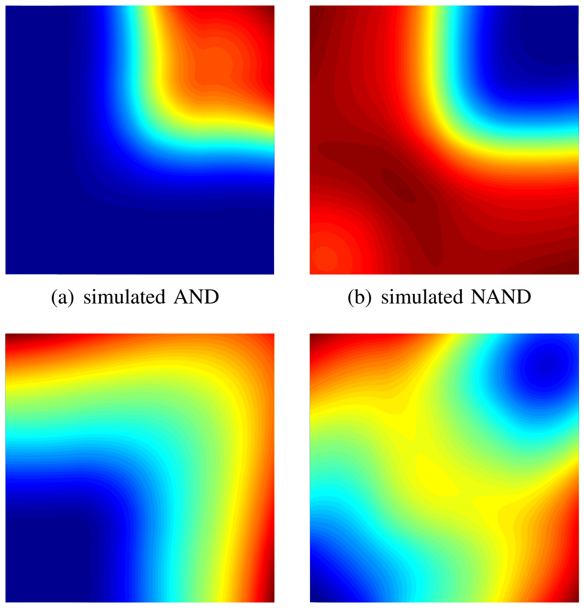


Figure 3. Contour plots of simulated evolved logic in the 4×4-grid

2) Discussion: From an electrical engineering point of view, the simulation results are very interesting, for several reasons. First of all, the example of Figure 2 can be configured into any of the basic Boolean logic gates, using only 16 nano-particles of size 5-20 nm. If one would like to design and build the same functionality with transistors, one would require at least 10 transistors. Secondly, in the designed circuit one would have to rewire the input and apply it at different places, whereas in the nano-particle network each of the input signals is applied at exactly one place. Even with current transistor sizes below 20 nm, the designed circuit would require the same or more space, and would dissipate substantially more energy.

It is interesting to note, that in the experiments with the real material samples [3], all basic Boolean gates were also evolved within an area with a diameter of around 200 nm, but with only six control voltages. However, currently these samples consist of 100-150 particles that are self-assembled into a disordered network. The experimental results as well as the simulations show the great potential for the approach, both in the bottom-up and top-down design regime. This could have a huge impact on future computing, either as an alternative approach to digital computing or as an alternative way to produce reconfigurable multi-functional devices, e.g., to support further down-scaling of digital components. Currently, we are not aware of any production techniques for constructing samples that come anywhere close to the 4×4-grid structure of Figure 2.

To obtain more insight in the underlying currents and physical phenomena, and with the long term goal to fully understand what is going on in terms of electron jumps and currents through these nano-particle networks, in [7] visualisation tools have been developed to analyse the processes that are taking place over time. In Figure 4, some pictures visualising the currents through the network are presented, in this case, as an example, when the network was configured as an AND. The amplitude of the currents is proportional to the area of the red arrows in the figure. The currents are all averaged over time. The tool also enables the calculation of variances, and it can show fast animations of the electron jumps as well. It is still an open problem to deduce an explanation for the patterns and jumps that cause the 4×4-grid to behave as a logic AND (or one of the other basic Boolean logic gates).

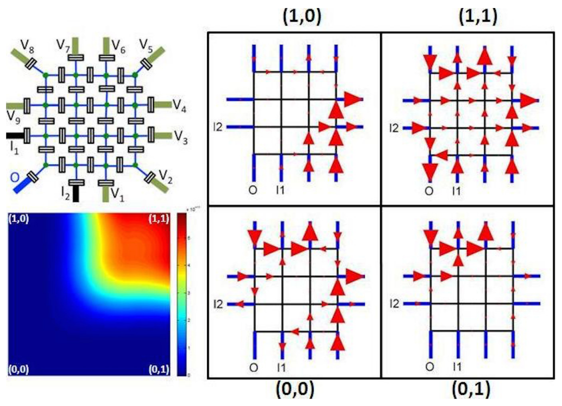


Figure 4. Averaged current patterns for simulated AND in the 4×4-grid

Figure 4 gives an impression of how complicated the traffic of electrons in such networks can get, and can hopefully in the future lead to more insight as to why they behave as logic.

In the next two sections, the use of artificial NNs to simulate the nano-material will be explained, as well as how to use the data collected from the above physical-model based simulations to explore the 4×4-grid.

IV. Neural Networks

Deep feed-forward NNs are powerful function approximators, and recently they have been very successfully applied in a wide range of domains. They consist of a sequence of layers, where each layer computes an affine projection of its inputs followed by a pointwise non-linear function ψ:

$$\mathbf{h} = \psi(\mathbf{W}\mathbf{x} + \mathbf{b}),$$

where $\theta = \{\mathbf{W}, \mathbf{b}\}$ are the parameters of the layer.

By stacking these layers, one can build non-linear functions of varying expressiveness that are differentiable. In theory, these networks can approximate any function to arbitrary precision given enough hidden units, and they also work very well in practice. This motivates the choice to use such deep feed-forward NNs to model the input-output characteristics of the nano-particle networks that were described earlier. These NNs are used in the sequel for approximating the mapping from the input and configuration voltages to the output current, by training them using many randomly chosen examples, generated with the physical-model based simulation tool. By solving that task for a specific nano-particle network, the trained NN becomes a differentiable model for the complex quantum-mechanic interactions within the material sample.

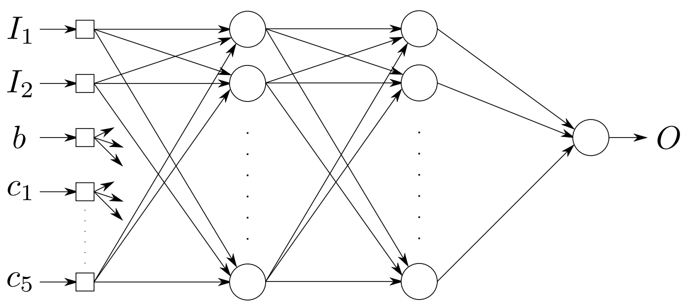


Figure 5. Illustration of an NN with two hidden layers

The voltages on the configuration leads and the inputs are scaled to have zero mean and unit variance, and serve as inputs to the NN. Referring to the 4×4-grid of Figure 2, in Figure 5, I_1 and I_2 denote the two input leads, b denotes the back gate, and $c_1, c_2, \ldots, c_5$ denote the other leads, in a symmetric fashion, so c_1 represents V_1 and V_9, and so on, whereas O denotes the output lead. The training objective is to minimise the Mean Squared Error (MSE) on the (also standardised) current of the output. The optimisation is done using Minibatch Stochastic Gradient Descent with Nesterov-style momentum [9] [12]. The Minibatch Stochastic Gradient Descent method is the de facto standard for optimising deep feed-forward NNs, whereas Nesterov-style momentum is a regularly used extension for improving convergence speed. There are other more elaborate optimisation schemes (like AdaGrad, RMSProp, ADAM, etc.), but their benefits are usually moderate, and there was no need for this added complexity here.

The choice of network architecture and training parameters comprise a set of hyperparameters that need to be set for this method. For the network, these are the number of hidden layers, the number of neurons in each layer and their activation functions. For training, they are the batch size, learning rate, the momentum coefficient, and stopping criterion. We fixed the batch size to 20, the momentum to 0.9, and stopped the training once the MSE on the validation set did not decrease for 5 epochs, or after a maximum of 100 epochs.

The hyperparameters of the NNs were optimised by random search over 40 runs [1] [15] [2]. In particular, we randomly sampled:

- learning rate η log-uniform from $[10^{-4}, 10^{-1}]$
- number of hidden layers from $\{1, 2, 5, 10\}$
- number of units in each layer from $\{8, 16, 32, 64, 128\}$
- activation function from $\{\text{ReL}, \tanh, \text{sigmoid}\}$.

The best NN had two hidden layers with 128 rectified linear units each, and was trained with a learning rate of $\eta \approx 1.6E{-}2$. This is the NN we use for the rest of the analysis.

A. *Search*

The trained NN is a differentiable (approximate) model of the nano-material. This property can be used to run the model "backwards": find inputs that produce certain desired outputs by using the backpropagation algorithm to perform gradient descent, not on the weights but on the inputs. Here, it was required to go even further and use backpropagation to search for functions; in particular, the aim is to find settings of the configuration leads such that various combinations of the input leads (logic pairs) produce corresponding desired (logic) outputs.

1) Local Search: To accomplish the above goal, it was necessary to produce a set of examples that have different values for the input leads but share the same (random) values for the configuration leads, along with the desired output values. Gradient descent is then used to minimise the MSE by adjusting the values for the configuration leads, while keeping their values the same for all examples. So formally, given our neural network model $\hat{f}$ of the nano-material, we define an error over our N input/output pairs $((I_1^{(i)}, I_2^{(i)}, O^{(i)}))$:

$$E = \sum_{i=1}^{N} \frac{1}{2}(\hat{f}(I_1^{(i)}, I_2^{(i)}, \theta) - O^{(i)})^2,$$

and then, using backpropagation, we effectively calculate:

$$\frac{\partial E}{\partial \theta} = (\hat{f}(I_1^{(i)}, I_2^{(i)}, \theta) - O^{(i)})\frac{\partial \hat{f}}{\partial \theta}.$$

2) Global Optimisation: One problem with the method described above, is that it only performs a local search, which means that the solution it converges to might correspond to a bad local minimum. To mitigate this, it was decided to first sample 10,000 random starting points (settings of the configuration leads), and perform just 10 iterations of the described local search on them. Only the starting point that leads to the lowest error is then optimised further for 5,000 epochs, in order to obtain the final solution. In this way, we reduce the risk of getting stuck in a poor local minimum. Each search comprises 420K evaluations of the neural network, for a total of about a minute on a modern CPU.

V. Results

In this section, we present the results of a network trained on 1M random function-evaluations collected from the simulated material. First of all, the following issue that was encountered and is illustrated in Figure 6 had to be resolved.

In Figure 6, only the 2,000 samples with the highest prediction error are shown. It can clearly be seen that the last ≈ 500 samples account for most of the total error. Also their target values are obviously outliers.

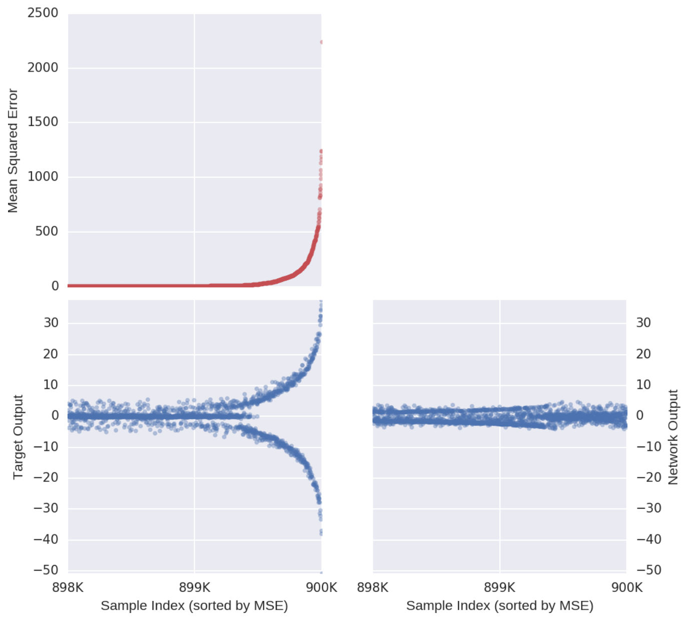


Figure 6. Prediction errors (top left), target outputs (bottom left), and network predictions (bottom right) for the simulated 4×4 nano-grid data

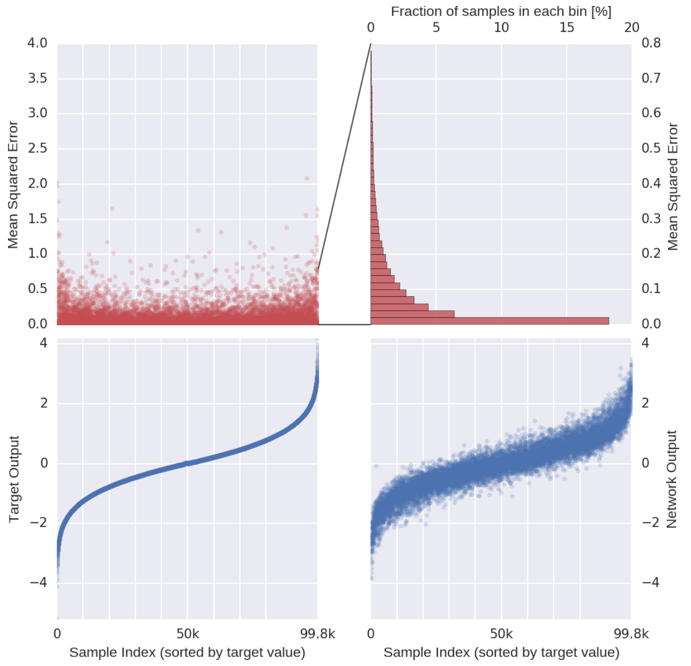


Figure 7. Prediction errors plus histogram (top), target outputs (bottom left), and network predictions (bottom right) for the *cleaned* nano-grid data

A. *Removing Outliers*

After training several NNs on the simulation data, it was discovered that more than half of the total prediction error stems from less than 0.1% of the data. These samples, which account for most of the error, also turned out to be outliers in terms of their output values, as can be concluded from Figure 6. Most of the data falls in the region $[-3, 3]$, but these 'bad' samples go up to ± 40. The gradient for training the NN is thus dominated by these few samples, causing the model to ignore the bulk of the data. For these reasons, it was decided to remove those outliers entirely from the dataset. After that, the NN performed much better in modelling the material, as can be seen in Figure 7. There, the samples are sorted by target value. One can observe a clear correspondence between the predictions (bottom right) and the targets (bottom left). The histogram in the top right confirms that most of the predictive errors are very small (< 0.3).

So, as a consequence, all results in the sequel have been obtained from the 'cleaned' data.

B. Logic Gates

The overall aim was to find configurations for the simulated 4×4 nano-grid that turns it into a multi-functional reconfigurable device for computing some well-known Boolean logic functions, just like the physical-model based simulation did: AND, OR, XOR, NAND, NOR, XNOR. For this goal, we first had to decide which values of (x_0, x_1) to map (False, True) to. The obvious choice is $(0, 1)$, but values of $(0.2, 0.8)$, for example, would also still be acceptable. To circumvent the problem of choosing these values, the same gradient descent method was used to adjust the values for True and False for the inputs as well.

In particular, the following was done for each function:

1) generate eight random numbers, while assuring that $x_1 > x_0$
2) using these values, create a set of four input/output pairs (see Table I)
3) perform gradient descent on these 8 values, while maintaining $x_1 > x_0$

The scheme by which the four input/output pairs are created from the eight random values $x_0, x_1, \ldots, x_7$ for any of the logic functions, is explained in Table I, in this case for the logic OR. Note that this depends on $x_0 < x_1$.

TABLE I. THE SCHEME FOR CREATING THE FOUR INPUT/OUTPUT PAIRS (FOR THE LOGIC OR)

I_1	I_2	c_1	c_2	c_3	c_4	c_5	b	out
x_0	x_0	x_2	x_3	x_4	x_5	x_6	x_7	OR(F, F)
x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7	OR(F, T)
x_1	x_0	x_2	x_3	x_4	x_5	x_6	x_7	OR(T, F)
x_1	x_1	x_2	x_3	x_4	x_5	x_6	x_7	OR(T, T)

As described in Section IV-A2, first a global search is performed, in order to find a good starting point, and then that vector is optimised further. The results are presented in Figure 8. There, the resulting configurations for six logic functions are illustrated (one logic function per row). The leftmost column shows the desired output for the logic function, while the middle column presents the actual response of the trained NN model. In the rightmost column, the corresponding configurations are visualised.

From the plots in Figure 8, it can be concluded that most of the target Boolean functions can be performed by the nano-material (according to the NN model). Note that, what is most important is the response of the model in the corners, since we do not really care about values in between True and False. The smooth plots are just there to show how the model is behaving, and to give an impression on how robust the solutions are. For AND, OR, NAND, and NOR, the values in the corners match the desired outputs very well. In contrast, the search

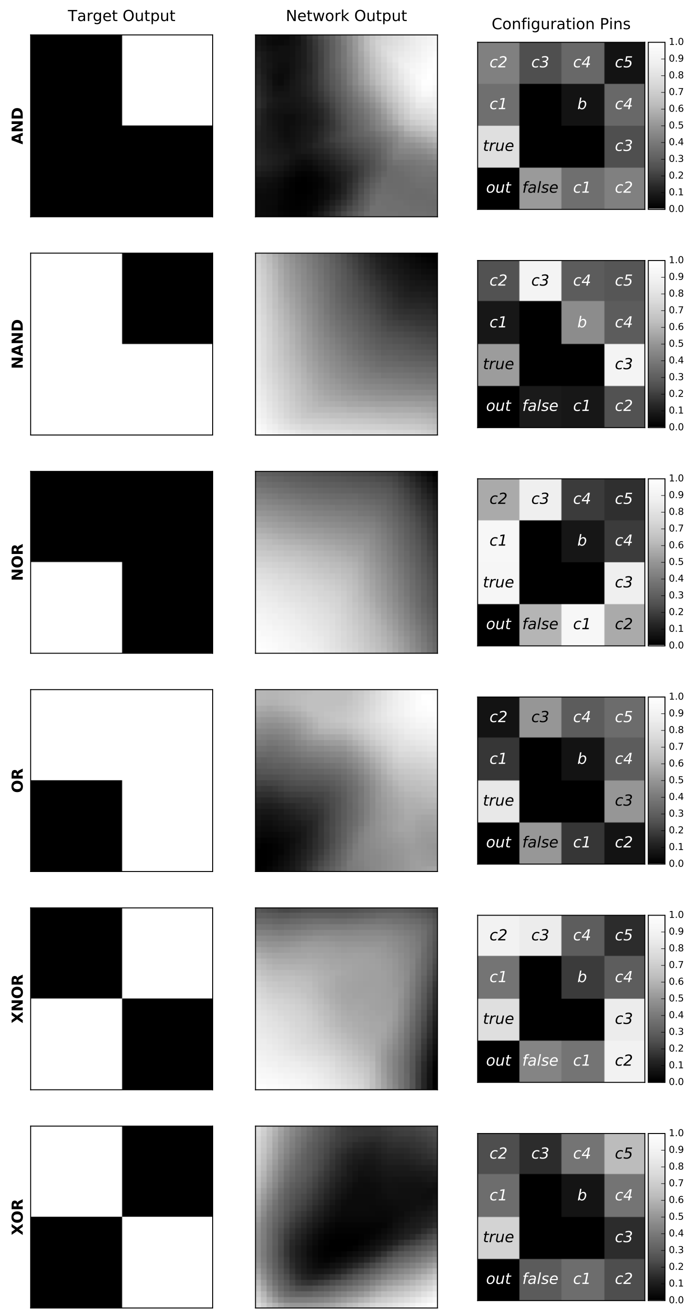


Figure 8. Results for six logic functions: desired outputs, actual responses, and the corresponding configuration

for the XOR and XNOR functions failed to produce equally satisfactory results, indicating that these functions might be difficult to perform for the nano-material under the given setup.

VI. Conclusion

This paper has demonstrated how an artificial Neural Network model can be applied to look for configuration voltage settings that enable different standard Boolean logic functions in the same piece of material consisting of a disordered nano-particle network. The training of the Neural Network was based on generated random data from a physical-model based simulation tool. The results are promising and can inform the electrical engineers about possible functional capabilities of these material systems, without the need of fabricating and doing costly and time-consuming trial-and-error experiments on real nano-particle networks. Of course, it is obvious that such experiments are unavoidable if it comes to actually testing real networks for the predicted functionalities. In fact, the capability of reconfigurable Boolean logic in small samples of nano-particle networks has recently been confirmed experimentally. It is likely, that this proof of concept will be the starting point for exciting new research, and open up the opportunity for a totally new approach to developing multi-functional stand-alone devices.

Next steps in this direction first of all involve the simulation of real material samples. For this, new experiments are needed, in order to produce sufficiently many data to enable proper training of the Neural Network. This also requires new fabrication techniques, involving larger networks on micro-electrode arrays, with more contact leads to the material, and with a more sophisticated back gate. The requirements can be predicted by simulations, in particular if one wants to turn to more complicated functionalities, like computational tasks that are difficult to perform with digital computers. Secondly, it would be worthwhile to apply the same modelling and simulation approach to other materials that show interesting physical properties and behaviour, like networks of quantum dots, films of carbon nanotubes, sheets of graphene, and mixtures of such materials. Thirdly, a natural next step would be to integrate the Neural Network modelling approach with the evolutionary search technique. These are amongst the future research plans we want to pursue.

ACKNOWLEDGMENT

We acknowledge financial support from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement No. 317662.

REFERENCES

[1] R. L. Anderson, "Recent advances in finding best operating conditions," *Journal of the American Statistical Association*, vol. 48, no. 264, pp. 789–798, Dec. 1953.

[2] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *The Journal of Machine Learning Research*, vol. 13, no. 1, pp. 281–305, 2012.

[3] S. K. Bose *et al.*, "Evolution of a designless nanoparticle network into reconfigurable boolean logic," *Nature Nanotechnology*, vol. 10, no. 12, pp. 1048–1052, 2015.

[4] H. Broersma, F. Gomez, J. Miller, M. Petty, and G. Tufte, "Nascence project: Nanoscale engineering for novel computation using evolution," *International Journal of Unconventional Computing*, vol. 8, no. 4, pp. 313–317, 2012.

[5] D. C. Ciresan, U. Meier, J. Masci, and J. Schmidhuber, "A committee of neural networks for traffic sign classification," in *International Joint Conference on Neural Networks (IJCNN)*, 2011, pp. 1918–1921.

[6] D. C. Ciresan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," in *IEEE Conference on Computer Vision and Pattern Recognition CVPR 2012*, 2012, long preprint arXiv:1202.2745v1 [cs.CV].

[7] R. M. J. van Damme, H. J. Broersma, J. Mikhal, C. P. Lawrence, and W. G. van der Wiel, "A simulation tool for evolving functionalities in disordered nanoparticle networks," *Preprint*, 2015.

[8] A. Korotkov, *Coulomb Blockade and Digital Single-Electron Devices*. Blackwell, Oxford, 1997, pp. 157–189.

[9] J. Martens and I. Sutskever, "Training deep and recurrent networks with hessian-free optimization," in *Neural Networks: Tricks of the Trade*. Springer-Verlag, 2012, pp. 479–535.

[10] W. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bulletin of Mathematical Biophysics*, vol. 7, pp. 115–133, 1943.

[11] L. Nagel and D. Pederson, "Simulation program with integrated circuit emphasis," University of California, Berkeley, Memorandum ERL-M382, April 1973.

[12] Y. Nesterov, "A method for unconstrained convex minimization problem with the rate of convergence o(1/k2)," *Doklady AN SSSR*, vol. 269, 1983.

[13] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain." *Psychological review*, vol. 65, no. 6, p. 386, 1958.

[14] H. Sak, A. W. Senior, and F. Beaufays, "Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition," *CoRR*, vol. abs/1402.1128, 2014, [retrieved: Feb, 2016]. [Online]. Available: http://arxiv.org/abs/1402.1128

[15] F. J. Solis and R. J.-B. Wets, "Minimization by random search techniques," *Mathematics of Operations Research*, vol. 6, no. 1, pp. 19–30, Feb. 1981.

[16] C. Wasshuber, *Computational Single-Electronics*. Springer-Verlag, 2001.

[17] ——, "Single-Electronics – How it works. How it's used. How it's simulated." in *Proceedings of the International Symposium on Quality Electronic Design*, 2012, pp. 502–507.

[18] C. Wasshuber, H. Kosina, and S. Selberherr, "A simulator for single-electron tunnel devices and circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 16, pp. 937–944, 1997.

[19] P. J. Werbos, "Applications of advances in nonlinear sensitivity analysis," in *Proceedings of the 10th IFIP Conference, 31.8 - 4.9, NYC*, 1981, pp. 762–770.

Towards a Fuzzy Logic Environment for Android

Ramón Luján
Dept. of Computing Systems, UCLM
Albacete (02071), Spain
Email: Ramon.Lujan@alu.uclm.es

Ginés Moreno
Dept. of Computing Systems, UCLM
Albacete (02071), Spain
Email: Gines.Moreno@uclm.es

Carlos Vázquez
Dept. of Computing Systems, UCLM
Albacete (02071), Spain
Email: Carlos.Vazquez@uclm.es

***Abstract*—In the recent past, several fuzzy extensions of the popular pure logic language Prolog have been designed in order to incorporate on its core new expressive resources for dealing with uncertainty in a natural way. Following this trail, during the last decade we have developed our Fuzzy LOgic Programming Environment for Research ($\mathcal{FLOPER}$) for providing a practical support to programs coded with an interesting flexible language emerged into the Fuzzy Logic Programming arena. As an example, our system has recently served us for developing a real-world application devoted to the flexible management of eXtensible Markup Language (XML) documents by means of a fuzzy variant of the popular XPath language. Nowadays, $\mathcal{FLOPER}$ is useful on computer platforms for compiling (to standard Prolog code), executing and debugging (by drawing execution trees) fuzzy programs, and it is ready for being extended in the near future with powerful transformation and optimization techniques designed in our research group during the last five years. In order to increase the portability of the system, in this paper, we initiate a research path devoted to accommodate its core on Android platforms. Nowadays, the environment accepts lattices modeling truth-degrees beyond the simpler crisp case {true; false} together with a wide range of user-defined fuzzy connectives for manipulating such truth degree. Moreover, the tool is able to manage fuzzy program rules whose syntax is very close to the one of Prolog clauses but admitting on their bodies elements coming from the lattice of truth degrees. Executing fuzzy programs into an Android environment is now possible after using our tool for compiling the fuzzy code to standard Prolog clauses and then using any one of the currently available Prolog interpreters for Android systems.**

Keywords–Fuzzy Logic Programming: Android; Compilers.

I. INTRODUCTION

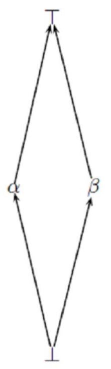
Fuzzy logic programming is the area of computing intended to introduce fuzzy logic [1] into logic programming [2], i.e., to formally address the vagueness and imprecision found in human reasoning. The DEC-Tau research group of the University of Castilla-La Mancha is immersed on this topic since 2006, when the first prototype of $\mathcal{FLOPER}$ was developed (see [3], [4], and [5]). $\mathcal{FLOPER}$ is able to interpret a very general fuzzy logic language called Fuzzy Aggregators and Similarity into a Logic Language (FASILL), as described in [6] and [7]. $\mathcal{FLOPER}$ has been enterely developed using the Prolog language (see Figure 1), which it is a natural basis for supporting fuzzy logic programming. Figure 2 shows an screenshot of the graphical interface of $\mathcal{FLOPER}$. We wish to remark here that, as illustrated in Figure 3, the system has served us for developing the so called FuzzyXPath language [14] (both the interpreter and the debugger associated to this real-world application can be freely downloaded an even tested on-line via the web page presented in [15]).

The $\mathcal{FLOPER}$ tool was originally thought to be executed in the environment of a personal computer or an equivalent machine. However, as time goes on mobile devices have spread and made computation ubiquitous. Our objective now is to update $\mathcal{FLOPER}$ to be present in those devices.

In this paper, we adapt fuzzy logic programming to mobile devices (smartphones, tablets, etc.), by presenting a version of the experimental prototype of $\mathcal{FLOPER}$ for the Android operating system. The objectives of this work include: 1) to make known logic programming to an increasing number of people, and 2) to ease the development of fuzzy logic programs.

There is no standard fuzzy logic programming language. Furthermore, there are two main trends (not necessarily opposing) to fuzzify logic programming (and, in particular, the standard logic programming language, Prolog):

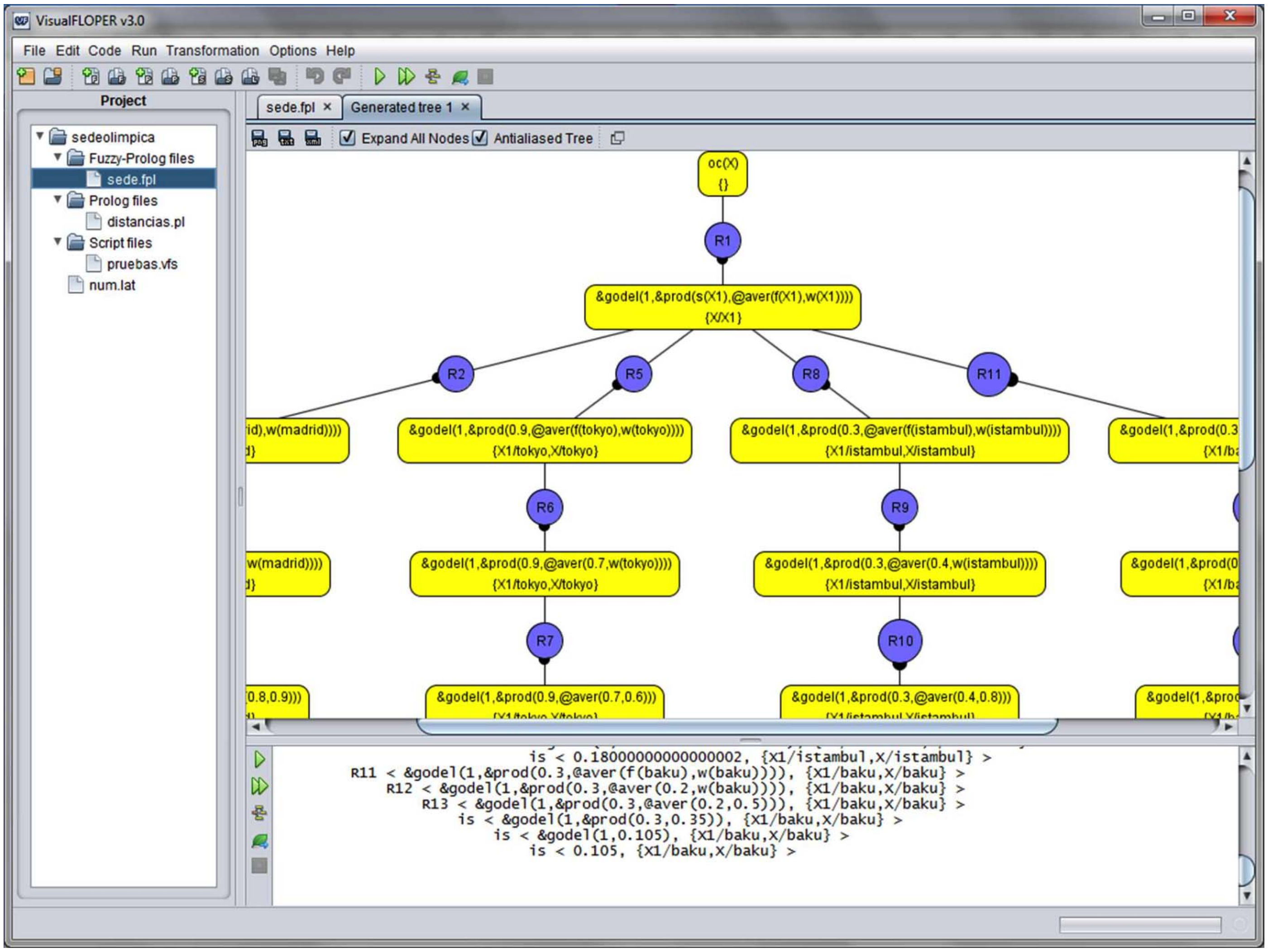
1) One family, that includes languages like Likeness in Logic (LIKELOG) [8], replaces the syntactic unification mechanism by some fuzzy unification algorithm. This unification algorithm can be based on different theoretical frameworks, of which one of the most successful is the use of similarity relations. In this family of languages, resolution remains basically unaltered, while unification is modified to consider that two different symbols (both predicates or both functions of the same arity) are “equal” at some degree defined by the established similarity relation. In this framework, programs are accompanied by a definition of a similarity relation, that may be in the form of a set of similarity equations, as is the case of BOUSI∼PROLOG [9], developed in the DEC-Tau group.



```
member(bottom).       member(alpha).
member(beta).         member(top).
leq(bottom,X).        leq(X,top).          leq(X,X).
top(top).             bot(bottom).
members([bottom, alpha, beta, top]).

and_godel(X,Y,Z)           :- pri_inf(X,Y,Z).
pri_inf(bottom,X,bottom)   :- !.
pri_inf(top,X,X)           :- !.
pri_inf(alpha,X,alpha)     :- leq(alpha,X),!.
pri_inf(beta,X,beta)       :- leq(beta,X),!.
pri_inf(X,Y,bottom).
```

Figure 1. Lattice of truth degrees modeled in Prolog for being used by the $\mathcal{FLOPER}$ environment.


VisualFLOPER v3.0
File Edit Code Run Transformation Options Help
Project
sedeolimpica
Fuzzy-Prolog files
sede.fpl
Prolog files
distancias.pl
Script files
pruebas.vfs
num.lat
sede.fpl
Generated tree 1
Expand All Nodes
Antialiased Tree
oc(X)
{}
R1
&godel(1,&prod(s(X1),@aver(f(X1),w(X1))))
{X/X1}
R2
R5
R8
R11
&godel(1,&prod(0.9,@aver(f(tokyo),w(tokyo))))
{X1/tokyo,X/tokyo}
&godel(1,&prod(0.3,@aver(f(istambul),w(istambul))))
{X1/istambul,X/istambul}
R6
R9
&godel(1,&prod(0.9,@aver(0.7,w(tokyo))))
{X1/tokyo,X/tokyo}
&godel(1,&prod(0.3,@aver(0.4,w(istambul))))
{X1/istambul,X/istambul}
R7
R10
&godel(1,&prod(0.9,@aver(0.7,0.6)))
&godel(1,&prod(0.3,@aver(0.4,0.8)))

```
                          is < 0.18000000000000002, {X1/istambul,X/istambul} >
          R11 < &godel(1,&prod(0.3,@aver(f(baku),w(baku)))), {X1/baku,X/baku} >
              R12 < &godel(1,&prod(0.3,@aver(0.2,w(baku)))), {X1/baku,X/baku} >
                  R13 < &godel(1,&prod(0.3,@aver(0.2,0.5))), {X1/baku,X/baku} >
                      is < &godel(1,&prod(0.3,0.35)), {X1/baku,X/baku} >
                          is < &godel(1,0.105), {X1/baku,X/baku} >
                              is < 0.105, {X1/baku,X/baku} >
```

Figure 2. When installed on a computer, programmers can comfortably interact with the $\mathcal{FLOPER}$ system thanks to its improved graphical interface.

SLD-Resolution ("Selective Linear Definite clause resolution") + Fuzzy (similarity) unification

2) In the other approach, programs are fuzzy subsets of formulae, where the membership degree of a formula to the subset is called its *truth degree*, and represents the confident the program has on that rule. Truth degrees are propagated through the modified resolution principle, while the unification mechanism remains unaltered. One example of these languages is Multi-Adjoint Logic Programming (MALP) [10].

Fuzzy SLD-Resolution + (syntactic) unification

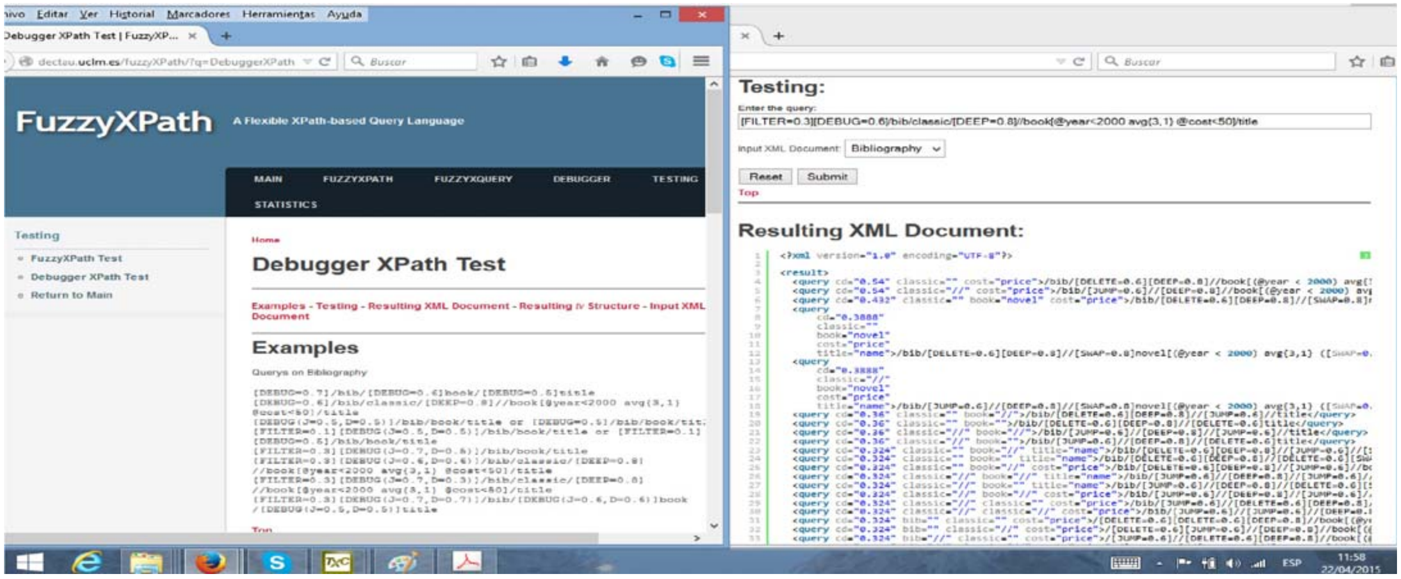
3) The last two approaches have been amalgamated in the form of the FASILL language, which modifies both the resolution principle and the unification mechanism of the original languages without introducing important changes on the final syntax.

Fuzzy SLD-Resolution + Fuzzy (similarity) unification

In this paper, we focus on the implementation of a programming environment based on fuzzy logic over Android, with the aims of compiling, visualizing, editing, creating and saving fuzzy logic programs in FASILL and their associated lattices of truth degrees. This environment has been implemented as an Android application using the Java language and the Eclipse Integrated Development Environment (IDE) in its kepler version.

The syntax of FASILL is similar to Prolog in the way functions and predicates are built, but it includes the advances of the multi-adjoint logic. Each program rule is accompanied by a truth degree that is an element of the associated lattice. Rules have an atomic head and a body, which is built up from atoms and truth degrees linked by connectives. Those connectives can be conjunctions, disjunctions or hybrid operators called aggregators, and can be defined by the logics of Gödel, Łukasiewicz, Product, or any other logic defined by the programmer. Furthermore, notice that all that is required for truth degrees is that belong to the multi-adjoint lattice associated to the program, so this framework goes beyond the $[0, 1]$ domain that limits the majority of fuzzy logic languages (see again Figure 1).

In this work, we provide a solution based on three main classes: a lexical analyzer for the new language containing the definition of its lexical categories; a syntactic analyzer that parses the grammar of FASILL; and a translator that produces


ivo Editar Ver Historial Marcadores Herramientas Ayuda
Debugger XPath Test | FuzzyXP...
dectau.uclm.es/fuzzyXPath/?q=DebuggerXPath
Buscar
FuzzyXPath
A Flexible XPath-based Query Language
MAIN
FUZZYXPATH
FUZZYXQUERY
DEBUGGER
TESTING
STATISTICS
Testing
FuzzyXPath Test
Debugger XPath Test
Return to Main
Home
Top
Reset
Submit
11:58
22/04/2015
ESP

Debugger XPath Test

Examples - Testing - Resulting XML Document - Resulting tv Structure - Input XML Document

Examples

Querys on Bibliography

```
[DEBUG=0.7]/bib/[DEBUG=0.6]book/[DEBUG=0.5]title
[DEBUG=0.6]/bib/classic/[DEEP=0.8]//book[@year<2000 avg{3,1}
@cost<50]/title
[DEBUG(J=0.5,D=0.5)]/bib/book/title or [DEBUG=0.5]/bib/book/tit
[FILTER=0.1][DEBUG(J=0.5,D=0.5)]/bib/book/title or [FILTER=0.1]
[DEBUG=0.5]/bib/book/title
[FILTER=0.3][DEBUG(J=0.7,D=0.8)]/bib/book/title
[FILTER=0.3][DEBUG(J=0.6,D=0.6)]/bib/classic/[DEEP=0.8]
//book[@year<2000 avg{3,1} @cost<50]/title
[FILTER=0.3][DEBUG(J=0.7,D=0.3)]/bib/classic/[DEEP=0.8]
//book[@year<2000 avg{3,1} @cost<50]/title
[FILTER=0.3][DEBUG(J=0.7,D=0.7)]/bib/[DEBUG(J=0.6,D=0.6)]book
/[DEBUG(J=0.5,D=0.5)]title
```

Testing:

Enter the query: [FILTER=0.3][DEBUG=0.6]/bib/classic/[DEEP=0.8]//book[@year<2000 avg{3,1} @cost<50]/title

Input XML Document: Bibliography

Resulting XML Document:

```
<?xml version="1.0" encoding="UTF-8"?>

<result>
  <query cd="0.54" classic="" cost="price">/bib/[DELETE=0.6][DEEP=0.8]//book[(@year < 2000) avg{
  <query cd="0.54" classic="//" cost="price">/bib/[JUMP=0.6]//[DEEP=0.8]//book[(@year < 2000) av
  <query cd="0.432" classic="" book="novel" cost="price">/bib/[DELETE=0.6][DEEP=0.8]//[SWAP=0.8]
  <query
      cd="0.3888"
      classic=""
      book="novel"
      cost="price"
      title="name">/bib/[DELETE=0.6][DEEP=0.8]//[SWAP=0.8]novel[(@year < 2000) avg{3,1} ([SWAP=0.
  <query
      cd="0.3888"
      classic="//"
      book="novel"
      cost="price"
      title="name">/bib/[JUMP=0.6]//[DEEP=0.8]//[SWAP=0.8]novel[(@year < 2000) avg{3,1} ([SWAP=0.
  <query cd="0.36" classic="" book="//">/bib/[DELETE=0.6][DEEP=0.8]//[JUMP=0.6]//title</query>
  <query cd="0.36" classic="" book="">/bib/[DELETE=0.6][DEEP=0.8]//[DELETE=0.6]title</query>
  <query cd="0.36" classic="//" book="//">/bib/[JUMP=0.6]//[DEEP=0.8]//[JUMP=0.6]//title</query>
  <query cd="0.36" classic="//" book="">/bib/[JUMP=0.6]//[DEEP=0.8]//[DELETE=0.6]title</query>
  <query cd="0.324" classic="" book="//" title="name">/bib/[DELETE=0.6][DEEP=0.8]//[JUMP=0.6]//[
  <query cd="0.324" classic="" book="" title="name">/bib/[DELETE=0.6][DEEP=0.8]//[DELETE=0.6][SW
  <query cd="0.324" classic="" book="//" cost="price">/bib/[DELETE=0.6][DEEP=0.8]//[JUMP=0.6]//bo
  <query cd="0.324" classic="//" book="//" title="name">/bib/[JUMP=0.6]//[DEEP=0.8]//[JUMP=0.6]/
  <query cd="0.324" classic="//" book="" title="name">/bib/[JUMP=0.6]//[DEEP=0.8]//[DELETE=0.6][
  <query cd="0.324" classic="//" book="//" cost="price">/bib/[JUMP=0.6]//[DEEP=0.8]//[JUMP=0.6]/
  <query cd="0.324" classic="//" classic="" cost="price">/bib/[JUMP=0.6]//[DELETE=0.6][DEEP=0.8]
  <query cd="0.324" classic="//" classic="//" cost="price">/bib/[JUMP=0.6]//[JUMP=0.6]//[DEEP=0.
  <query cd="0.324" bib="" classic="" cost="price">/[DELETE=0.6][DELETE=0.6][DEEP=0.8]//book[(@y
  <query cd="0.324" bib="" classic="//" cost="price">/[DELETE=0.6][JUMP=0.6]//[DEEP=0.8]//book[(
  <query cd="0.324" bib="//" classic="" cost="price">/[JUMP=0.6]//[DELETE=0.6][DEEP=0.8]//book[(
```

Figure 3. An on-line session with the *FuzzyXPath* debugger developed with $\mathcal{FLOPER}$.

an object program from a source program following the rules that we detail in the next sections. As part of our goal, our platform have to report possible errors in the source program detected in the lexical or syntactic analysis. Those reports must clarify where in the code is the error, its type and also a possible cause of it.

The structure of this paper is as follows. Section II resumes the main elements involved in the construction of a compiler. Next, Section III details the Android-based implementation of our tool. Finally, in Section IV, we conclude by also proposing several lines for future research.

II. Language Processors

A compiler is a software tool such that one of its main input data is a language, as explained in [11] and [12]. There are many different kinds of language processors, including compilers and interpreters. Compilers are programs which can read a program written in a language (called the source language) to translate it to an equivalent program in another language (called the target language). The source program is usually a high-level language and the target language is usually a machine language.

Unlike a compiler, an interpreter is a language processor which executes a source program to return the result of that execution [11]. According to Louden [12] and Scott [13], the target machine language program that produces a compiler is usually faster than an interpreter at the moment of assigning the inputs to the outputs, because a decision made by compile time is a decision that does not need to be made at run time. However, an interpreter may provide better diagnoses because it executes the source program instruction by instruction.

Both of them, compilers and interpreters, are really complex tools. They may have around ten thousand lines of code [12]. Luckily, in spite of this complexity, as the knowledge and the tools to structure them are known, the complexity is reduced. The tasks of these kind of language processors are explained below (see Figure 4):

1) Analysis: the analysis divides the program into components and imposes a grammatical structure on them. Then, it uses this structure to create an intermediate representation of the source program. If the analysis detects that the source program is malformed in terms of syntax or semantics, then it must provide messages for the user to correct it.
    a) Scannings: the scanner, or lexical analyzer, reads the stream of characters that make up the source program and join them together to produce tokens (each token is a character string representing a unit of information in the source program). These tokens are classified into categories, which are defined using regular expressions. Typical examples of categories are reserved words, identifiers, special symbols, constants, etc. When the lexical analyzer finishes its analysis, it sends those tokens to the parser in order to help it to analyze the structure of the program.
    b) Parsing: the parser, or syntax analyzer, uses the tokens and grammatical rules of a context-free grammar in order to build an intermediate representation in a tree which describes the grammatical structure of the flow of tokens. The most common representation is the Abstract Syntax Tree (AST), wherein each internal node represents an operation and its descendants represent the arguments of the operation. An example of the abstract syntax tree is seen in Figure 5.
    c) Semantic analysis: the semantic analyzer uses the syntax tree to check the semantic consistency of the program. It checks that semantic language restrictions are met. Some examples may be unable to add a character and an integer, or not to use a variable that has not been previously declared.

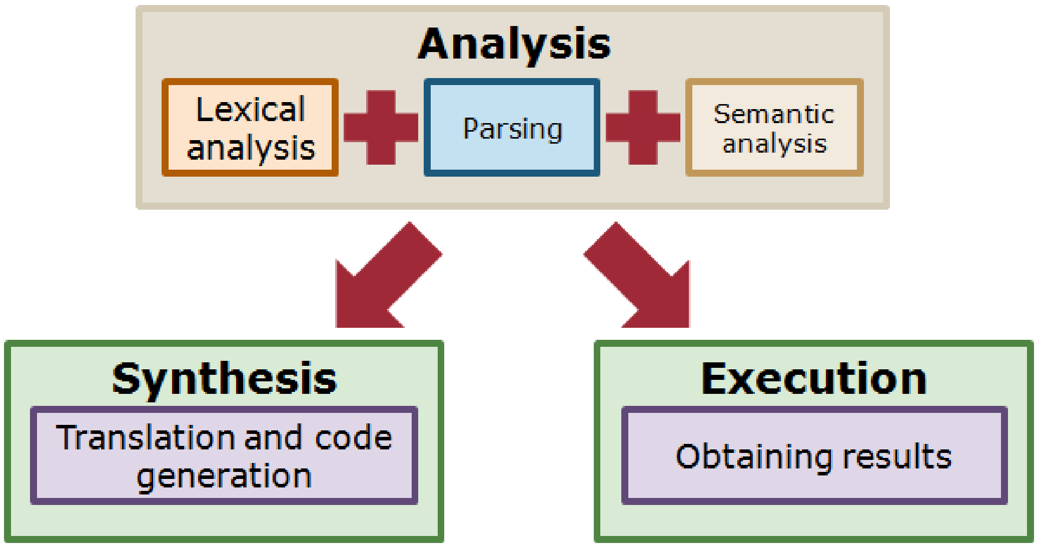


Figure 4. Translation stages.

2) Synthesis (compilers): it builds the desired target program using the intermediate representation.
3) Execution (interpreters): it executes the program using the intermediate representation and returns the results.

The aim of compilers is to obtain a translation from a source language to an object language, while the aim of interpreters consists only in the program execution and getting the outcomes (within this investigation a compiler has been developed, but the functions of an interpreter will be added in the following versions).

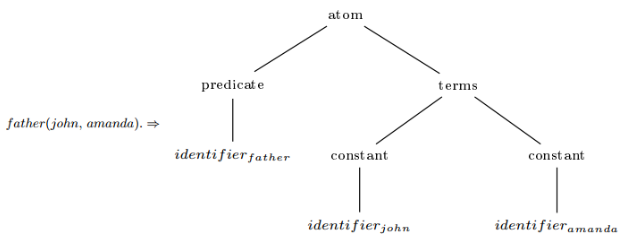


Figure 5. Abstract syntax tree.

III. Implementation Based on Android

In this section, we detail how this compiler was developed, explaining the implementation of the lexical analyzer and the parser in detail. It has not been necessary to implement a semantic analyzer because the grammar of this compiler only admits Horn clauses. But first, we briefly explain the main reasons why Android has been selected as the operating system and in particular Android tablets for the implementation of this compiler.

The first reason is that many people use this operating system in their daily life; it is also really easy to get a device that works with Android. In addition, tablets have been selected as the target device because of their advantages. A tablet offers better mobility than other devices such as computer towers and laptops. Also, tablets have a better display than mobiles,

TABLE I. SPECIFICATION OF LEXICAL CATEGORIES.

Lexical category	Pattern	Attributes
Identifier	[a-z][_a-zA-Z0-9]*	Lexeme, line and column
Variable	[A-Z][_a-zA-Z0-9]*	Lexeme, line and column
Number	[0-9]+(.[0-9]+)?	Lexeme, line and column
Comment	%(.)*	Lexeme, line and column
Connective	[@&|]	Lexeme, line and column
LPAREN	[(]	Line and column
RPAREN	[)]	Line and column
WHITE	[\t\r]	Line and column
NL	[\n]	Line and column
COMMA	[,]	Line and column
PERIOD	[.]	Line and column
FROM	(<-)	Line and column

which could improve the interactions between the user and the application in order to make them more comfortable. Finally, tablets are more powerful than mobiles, which is very useful for a programming environment.

Before proceeding with the implementation of the compiler in Android, it must be clarified that in this version of the compiler is the syntax analyzer which requests the tokens to the lexical analyzer when it needs them. This is known as a translation guided by the syntax.

A. *Analysis*

As it was mentioned in the section of Language Processors, this is the stage in which the AST corresponding to the source program is obtained.

Below we detail the development of the lexical analyzer and the parser, showing the lexical categories and the rules which have been implemented to the language of the compiler.

1) Lexical Analysis: As it was mentioned before, in this stage the input is split up in tokens to be clasified in lexical categories later. Also, these tokens contain attributes which could be useful to following stages.

Table I shows the different lexical categories which have the compiler. It is easy to see patterns or regular expressions, which are used to define them, besides the attributes of the categories. Line and column attributes are used to give

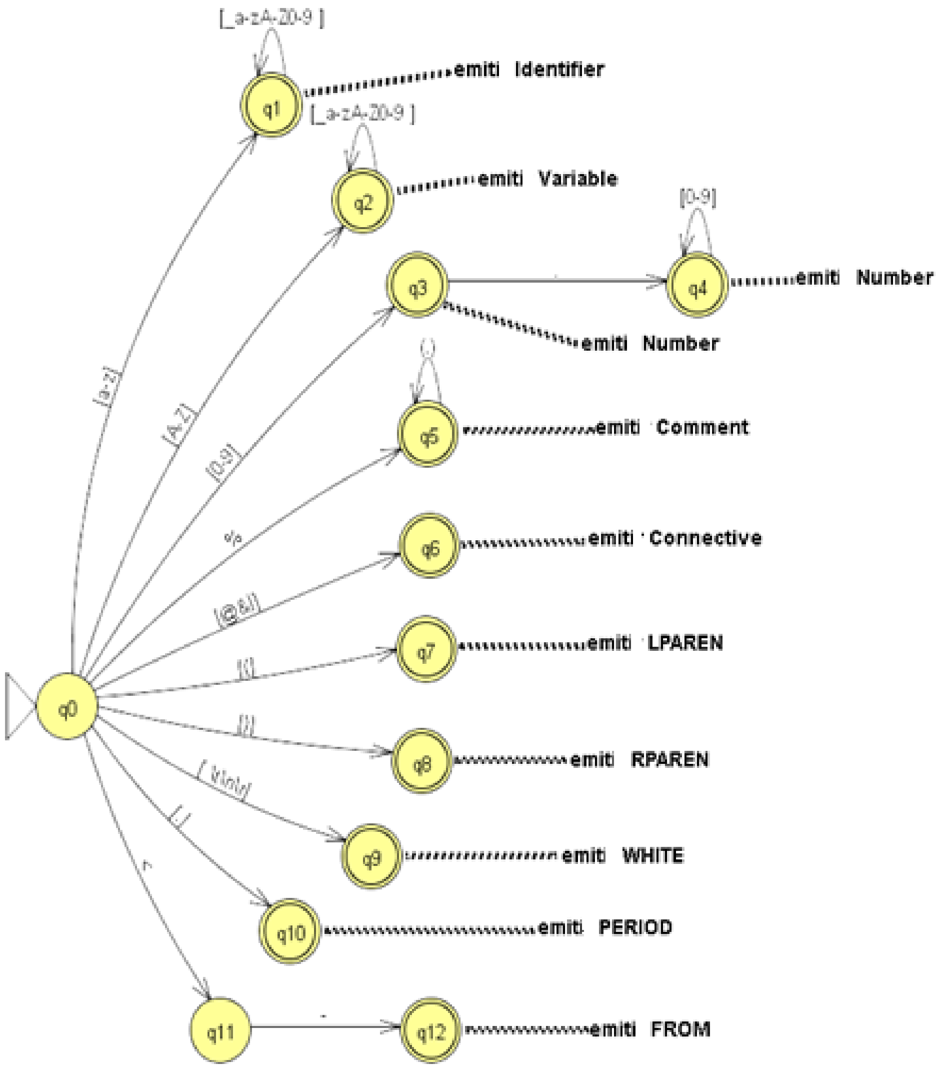


Figure 6. DDM associated to the language of the compiler.

information about the position of an error, either a lexical error or a syntax error, while the lexeme attribute is used to store the value of a token, which is used in the stage of synthesis.

To be able to perform the lexical analysis, a Deterministic Discriminating Machine (DDM) has been built. It splits the input up into individual characters and acts on them repeatedly, starting each time on a different point, but it always starts in its initial state.

The DDM follows a greedy strategy. This means that it searches the biggest token that belongs to a lexical category on the portion of input that has not been analyzed yet. Thus, splitting the input exactly matches that expected.

This DDM has been developed by Java code using the if-else sentence to compare patterns with the current character of the input. Below, the DDM associated to the language of the compiler will be shown in Figure 6.

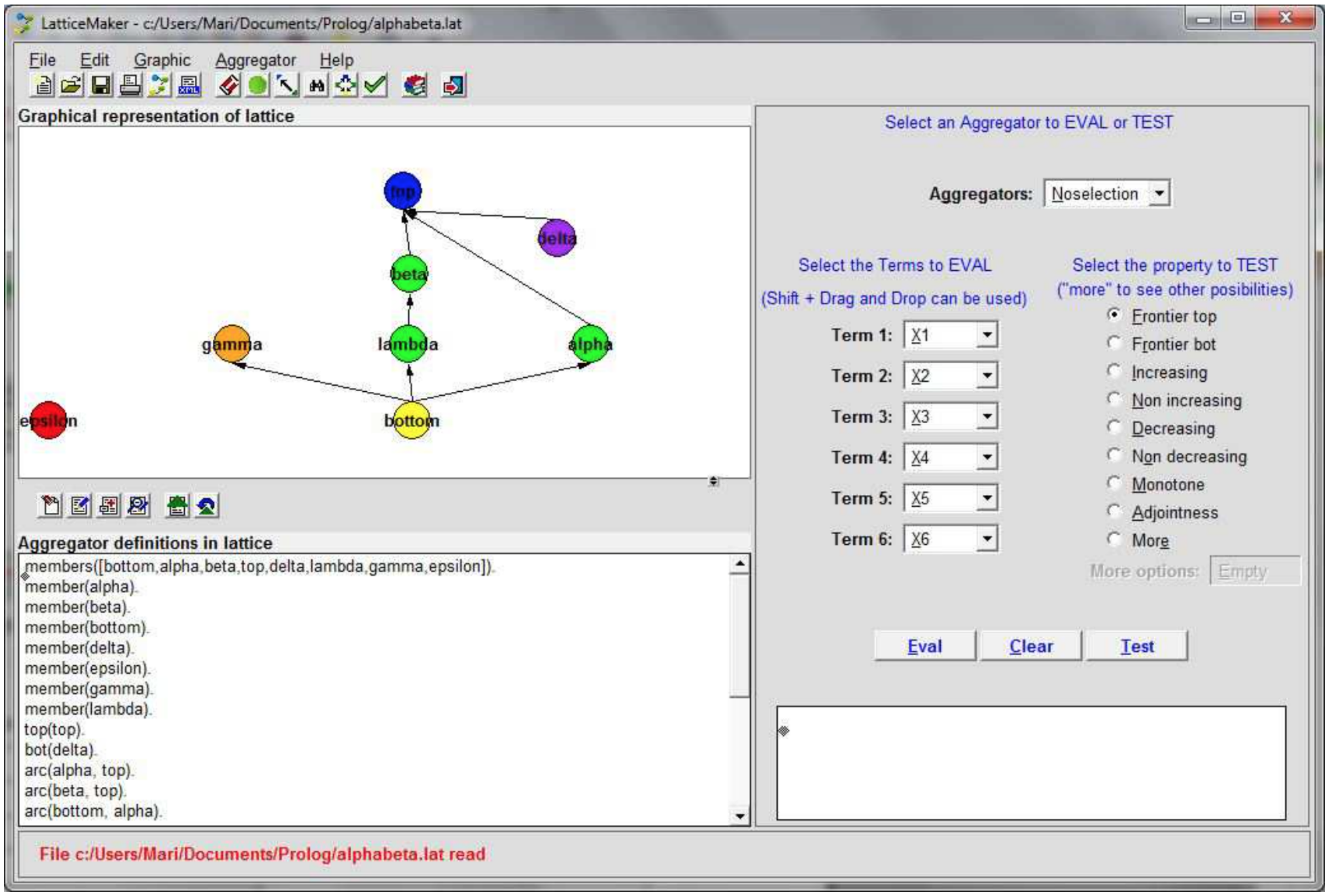
2) *Parsing:* As it was commented before, the syntax analyzer is in charge of discovering the structures of the code using context-free grammars and the tokens which are sent from the lexical analyzer. These grammars have enough expressive power to represent most of the constructs that are in the compiler. Also, they are easy to develop and lead to a very efficient analysis.

From tokens and the context-free grammar defined for this compiler, the syntax analyzer builds a parse tree, which contains information about how to make the input using grammatical rules.

The syntax analysis is a descendent process. This means that the tree is built from the root to the leaves. This process is needed to predict what the next tokens are going to be; this is why it is also known as predictive analysis.

Below is shown the grammar used to develop this compiler:

- $\langle Program \rangle$ = $\langle Line \rangle$ (NL $\langle Line \rangle$)*: programs are composed by one or more lines separated by new lines.
- $\langle Line \rangle$ = $\langle Rule \rangle$|$\langle Goal \rangle$|COMMENT: lines can be rules, goals or comments. Unlike other compilers, in this compiler comments are considered in the analysis because it has been considered important that the comments will appear in the translated program.
- $\langle Goal \rangle$ = GOAL $\langle Body \rangle$ PERIOD: despite of the application not having the functionality of an interpreter, it include the rules necessary to analyze goals.



Aggregators: Noselection

Term 1: X1
Term 2: X2
Term 3: X3
Term 4: X4
Term 5: X5
Term 6: X6

[✓ Frontier top]

More options: Empty

```
members([bottom,alpha,beta,top,delta,lambda,gamma,epsilon]).
member(alpha).
member(beta).
member(bottom).
member(delta).
member(epsilon).
member(gamma).
member(lambda).
top(top).
bot(delta).
arc(alpha, top).
arc(beta, top).
arc(bottom, alpha).
```

File c:/Users/Mari/Documents/Prolog/alphabeta.lat read

Figure 7. Screenshot of the *"LatticeMaker"* tool assisting the graphical design of a lattice of truth degrees.

- $\langle Rule \rangle$ = $\langle Head \rangle$ (FROM $\langle BODY \rangle$)? PERIOD: rules can be FASILL facts (rules without a body) or rules with a head and a body.
- $\langle Head \rangle$ = $\langle Atom \rangle$: a head is an atom.
- $\langle Atom \rangle$ = $\langle Predicate \rangle$ (LPAREN $\langle Terms \rangle$ RPAREN)?: an atom is a predicate which could contain a term list.
- $\langle Terms \rangle$ = $\langle Term \rangle$ (COMMA $\langle Term \rangle$)*: a term list consist on terms separated by commas.
- $\langle Term \rangle$ = IDENTIFIER (LPAREN $\langle Terms \rangle$ RPAREN)?|VARIABLE: a term can be a function (an identifier which in this case would be a function symbol followed of a term list), a variable or a constant (a single identifier).
- $\langle Body \rangle$ = $\langle Atom \rangle$ | $\langle TruthDegree \rangle$ | $\langle Connective \rangle$ LPAREN $\langle ArgumentList \rangle$ RPAREN: a body can be an atom, a truth degree or a connective with an argument list.
- $\langle TruthDegree \rangle$ = IDENTIFIER | NUMBER: a truth degree can be an identifier or a number, depending on how the lattice has been defined.
- $\langle Connective \rangle$ = CONNECTIVE IDENTIFIER?: connectives consist on a connective symbol (@, & o |) which can be followed of an identifier tag.
- $\langle ArgumentList \rangle$ = $\langle Body \rangle$ (COMMA $\langle Body \rangle$)*: an argument list consist on bodies separated by commas.

B. *Synthesis*

In this stage, the program with the FASILL syntax is translated to Prolog. A translation is generated from the tree made in the stage of analysis. These are the rules that have been used to generate the translation:

- Atoms: every atom written in FASILL is translated to Prolog by adding a truth degree. In some cases it will be a variable TVX (where X corresponds to an index depending on the position of the atom) which will contain the value and in other cases it will be the value.

  q(X, Y). ≡ q(X, Y, TV0).
- Facts: these atoms will be translated to Prolog by adding the maximum truth degree of the lattice, as a fact is always true in fuzzy logic.

  q(X, Y). ≡ q(X, Y, 1).
- Rules with body: there are two different cases.
  - The body is a truth degree: this case is very simple to treat, because it is translated as a Prolog fact with the truth degree added.

    q(X, Y) <- 0.6. ≡ q(X, Y, 0.6).
  - The body consists on a connective with arguments: in this case atoms will be translated in order, following the steps described above. The translation of the connectives consists on an

atom whose parameters are the variables of the truth degrees of the atoms which it contained, added to its own truth variable.

q(X, Y) <- &prod(0.6,p(X)). $\equiv$ q(X, Y, TV0) :- p(X,TV1), and_prod(0.6,TV1,TV0).

IV. Conclusions and Future Work

In this paper, we have described the first prototype of $\mathcal{FLOPER}$ for Android, which tries to facilitate the access to fuzzy logic programming to the people.

Our application is still in an early stage of development and can be improved in several ways, that will constitute our focus of attention. We plan to implement the similarity management present in the personal-computer version of $\mathcal{FLOPER}$, as well as an interpreter able to evaluate program goals. Furthermore, we intend to introduce a mechanism to allow the graphical design of lattices, which would greatly facilitate the creation and visualization of truth degree lattices (in Figure 7 we show a preliminary tool that we have recently developed in this sense for computers [16]).

Other possibilities concern system usability. We intend to improve the error detection system in both the lexical and syntactic levels, so as to provide more information related to errors. Also, since smartphone and tablet keyboards are neither easy nor comfortable to use, we intend to provide direct access to the most commonly used symbols in FASILL programs, like “<-”, “.”, “(“, “)”, etc. In this sense, we also plan to introduce word completion for identifiers and variables in order to gain ease of use.

Acknowlegments

This work was supported by the EU (FEDER), and the Spanish MINECO Ministry (*Ministerio de Economía y Competitividad*) under grant TIN2013-45732-C4-2-P.

References

[1] L. A. Zadeh, “Fuzzy logic and approximate reasoning,” Synthese, vol. 30, 1965, pp. 407–428.

[2] J. Lloyd, Foundations of Logic Programming. Springer-Verlag, Berlin, 1987, second edition.

[3] J. Abietar, P. Morcillo, and G. Moreno, “Designing a Software Tool for Fuzzy Logic Programming,” in Proc. of the ICCMSE’07 International Conference of Computational Methods in Sciences and Engineering, Corfu, Greece, September 25-30, T. Simos and G. Maroulis, Eds., vol. 2. American Institute of Physics, 2007, pp. 1117–1120, distributed by Springer Verlag. ISBN 978-0-7354-0478-6.

[4] G. Moreno and C. Vázquez, “Fuzzy logic programming in action with FLOPER,” Journal of Software Engineering and Applications, vol. 7, 2014, pp. 273–298.

[5] DEC-TAU research group (University of Castilla-La Mancha). The Fuzzy Logic Programming Environmment for Research FLOPER, web page at http://dectau.uclm.es/floper/?q=sim, 2015.

[6] G. Moreno, J. Penabad, and C. Vázquez, “Beyond multi-adjoint logic programming,” International Journal of Computer Mathematics, vol. 92, 2014, pp. 1956–1975.

[7] P. J. Iranzo, G. Moreno, J. Penabad, and C. Vázquez, “A fuzzy logic programming environment for managing similarity and truth degrees,” in Proceedings XIV Jornadas sobre Programación y Lenguajes, PROLE 2015, Cadiz, Spain, September 16-19, 2014., ser. EPTCS, S. Escobar, Ed., vol. 173, 2015, pp. 71–86. [Online]. Available: http://dx.doi.org/10.4204/EPTCS.173.6

[8] F. Arcelli and F. Formato, “Likelog: A logic programming language for flexible data retrieval,” in Proc. of the ACM Symposium on Applied Computing, SAC’99, San Antonio. ACM, Artificial Intelligence and Computational Logic, 1999, pp. 260–267.

[9] P. Julián and C. Rubio, “An efficient fuzzy unification method and its implementation into the Bousi$\sim$Prolog system,” in 2010 IEEE International Conference on Fuzzy Systems, FUZZ-IEEE’10, July 18-23, Barcelona, IEEE, Ed., 2010, pp. 658–665.

[10] J. Medina, M. Ojeda-Aciego, and P. Vojtáš, “Multi-adjoint logic programming with continuous semantics,” in Proc. of Logic Programming and Non-Monotonic Reasoning, LPNMR’01, Vienna, Lecture Notes in Artificial Intelligence 2173, 2001, pp. 351–364.

[11] A.V. Aho, M.S. Lam, R. Sethi, and J.D. Ullman, Compilers: Principles, Techniques, and Tools (2nd Edition). Addison Wesley, August 2006. [Online]. Available: http://www.amazon.ca/exec/obidos/redirect?tag=citeulike09-20&amp;path=ASIN/0321486811

[12] K. C. Louden, Compiler Construction: Principles and Practices. Boston, MA, USA: PWS Publishing Co., 1997.

[13] M. L. Scott, Programming Language Pragmatics. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2009.

[14] J. M. Almendros-Jiménez, A. L. Tedesqui, and G. Moreno, “Fuzzy xpath through fuzzy logic programming,” New Generation Computing, vol. 33, no. 2, 2015, pp. 173–209. [Online]. Available: http://dx.doi.org/10.1007/s00354-015-0201-y

[15] DEC-TAU research group (University of Castilla-La Mancha). The fuzzyXPath language, interpreter and debugger, web page at http://dectau.uclm.es/fuzzyXPath/, 2010.

[16] J. Guerrero, M. Martínez, G. Moreno, and C. Vázquez, “Designing lattices of truth degrees for fuzzy logic programming environments,” in 2015 IEEE Symposium Series on Computational Intelligence: Symposium on Foundations of Computational Intelligence (2015 IEEE FOCI), Cape Town, South Africa, 2015, pp. 10 (In press, IEEE).

Evaluation of Jurisprudence Arguments Based on Annotations of Logical Structures and Speech Acts

Shumpei Kubosawa, Shogo Okada and Katsumi Nitta

Interdisciplinary Graduate School of Science and Engineering
Tokyo Institute of Technology
4259 Nagatsuta-cho, Midori-ku, Yokohama, Japan 226–8502
Email: kubosawa@ntt.dis.titech.ac.jp, {okada,nitta}@dis.titech.ac.jp

Abstract—**In jurisprudence education, discussion trainings by mock arbitrations, mock mediations and so on are conducted. On those trainings, students discuss an issue and experienced supervisors evaluate the discussion by observing them and make comments with respect to argumentation and other contextual abilities. However, evaluation consume much time of experts, and objective criteria for evaluation are not established yet. Besides, existing tools for supporting them are insufficient to analyze a total discussion log and compare them. Furthermore, there are hardly any objective method to analyze contextual abilities. In order to support evaluation of actual discussion records, we propose a method and supporting tools for discussion evaluation. The presented method is based on annotations of logical structure of total arguments and speech acts attached to each utterance on discussion logs. We compared evaluation result of our method to expert's manual evaluation and showed that evaluation results by our method was similar to experts' result.**

***Keywords*–argument analysis; jurisprudence education; argumentation framework; Toulmin model.**

I. Introduction

In education for students majoring in law, argumentation training through mock trials or arbitration is introduced with the goal of enhancing their operational capability of law and arbitration skills. In this type of training, the participants proceed with a conflict-case discussion as the parties concerned. The teacher evaluate the discussion and gives them counsel with respect to argumentation and discussion skills after the discussion. For the purpose of giving more opportunities to make comments to as many students as possible, recently, online discussion training programs have been introduced by utilizing electronic bulletin boards.

In addition to this kind of training, argumentation competitions including Intercollegiate Negotiation Competition (INC) [1] are held in order to bring about further enhancement of argumentation skills. They develop discussions in two forms, a mock arbitration and a mock negotiation, conducted in about 20 different locations. The evaluation items for each discussion are described on a score sheet. At each location, legal experts score and compare these evaluation items in order to evaluate and rank each team.

This type of arbitration training requires that a limited number of teachers need to observe and evaluate a number of discussions. Not only observe each individual discussion in detail, but they also need to compare multiple discussions in order to evaluate the relative merits. Therefore, teachers are placed under significant burden and pressure, while support based on computers becomes very necessary.

However, the focus of previous studies on discussion record analysis (described in Section 2) remains to have only partial support for analysis, such as the visualization of argument structures. These methods are insufficient for determining the discussion quality and characteristics. Furthermore, objective criteria for evaluation of argumentation and contextual abilities are not established yet. Contextual abilities cover attitudes of mediators, smoothness of discussions, deadlock response, and other skills. For instance, one of the important contextual skills for mediators is reframing, while workshops targeting settlement committees have been conducted. Reframing is a type of utterance which can potentially resolve deadlocks. In spite of the importance, there are hardly any objective analysis with respect to reframing in conventional studies to our knowledge. Therefore, those works have not met the needs for discussion training support.

We introduce a novel method and a support tool for discussion analysis. In order to calculate proposing evaluation indices and detect deadlocks for supporting reframing analysis, our method utilize sequence of tags attached to each utterance in discussion record. Logical structure tags and speech act tags mainly correspond to argument structure analysis and discussion skill analysis respectively. Section 2 describes related studies regarding discussion analysis. Section 3 handles the outline of the discussion analysis support tool, while Section 4 shows the discussion analysis results obtained by using this tool and shows that the result is similar to experts' evaluations.

II. Related work

A. Plotting of discussion structures and tagging

Many researchers have studied how to analyze logical structures of arbitration by visualizing them based on diagrams. For example, Toulmin proposed the model (Toulmin model) which visualizes arguments as diagrams by dividing them into each element of claim, data, warrant, backing, qualifier, and rebuttal [2] (Figure 1). The Toulmin model is commonly known among legal experts, while it is widely accepted that sound arguments can be divided into these elements.

In addition, there have been developed many systems based on Toulmin-model-based diagrams, such as an online system that supports discussions while presenting discussion

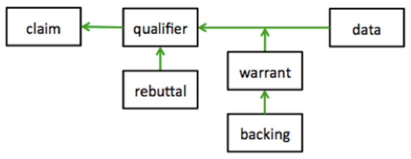


Figure 1. Toulmin model

structures on a real-time basis, and a system that analyzes logical structures of a discussion [3][4][5][6].

Furthermore, in order to extract logical structures from discussion records, tagging the role of each individual word, section, or paragraph are useful. The concept of Global Document Annotation (GDA) proposed by Hashida et al. provides a universal tagging method [7]. The tagging method proposed by Tanaka et al. categorizes the roles of each utterance into claims, agreements, denials, complements, reasons, withdrawals, questions, answers, proposals, requests for reasons, and transfer of the right. Tagged utterances are then used for automatic extraction of the factors of argument and the search for similar discussion scenes [8].

B. Argumentation frameworks

An argumentation framework is a framework that is provided in order to select valid arguments based on attack relationships between arguments. Dung defined the possibility of accepting each argument within a discussion as the principle that "The one who has the last word laughs best." Namely, he defines this possibility as a dialogical relationship based on the concept that "although argument A is attacked by argument B, if argument B attacks (countercharges) argument A, argument A is not defeated by argument B." Based on this definition, he proposed the following argumentation frameworks (AF) [9].

$$AF = (Arg, attacks) \qquad (1)$$

The AF (1) is a pair consisting of a set of arguments Arg and $attacks \subseteq Arg \times Arg$, which indicates the attack relationships between arguments. In the AF, the possibility of accepting each argument is evaluated based on the attack relationships between arguments and multiple semantics are defined.

Prakken proposed the framework referred to as ASPIC+ framework which incorporated logical expressions into argumentation frameworks [10]. This framework provides a method to interpret discussion structures based on AF by transforming argument trees consisting of strict rules and defeasible rules into AF.

C. Discussion analysis based on points of dispute

In the field of support for argumentation education, Ashley and Aleven proposed discussion models based on exchanges of legal precedents which were expressed as a list of points (dimensions or factors) of dispute [11][12]. The method proposed by Ashley et al. analyzes those sets of precedents that are preliminarily available in order to extract dimensions or factors that characterize individual precedents in advance.

Expanding this method, Tanaka et al. developed an automatic method to recognize points of dispute by recognizing the factors included in individual comments made during a discussion based on pre-prepared lists of factors [8].

D. Reframing

Reframing is one of the discussion skills required of the committee of settlement in an arbitration case [13]. Reframing is a type of utterances that can change the frame of thinking of other parties. This is done in order to change the thought of them from negative things to positive things, or in order to persuade them [14]. Reframing is an effective technique for arbitrations when the discussion stagnates and seems to go nowhere due to emotional reactions expressed by both parties or other reasons. However, it is difficult to decide what to say and when to say for reframing since they depend on the context of discussion. Therefore, automated detection of reframing portins in a discussion record is also difficult, while reframing utterances would tend to exist after deadlocks by its nature.

III. Discussion Evaluation Methods and an Evaluation Support Tool, Corte

A. Outline of Corte

The purpose of our discussion analysis support is to support discussion education in the law schools of universities as described in Section 1. Those jurisprudence discussions have three characteristics: (1) facts on the disputed case are preliminarily given, (2) factors correspond to each claim are predicted in advance, and (3) there are multiple discussion records regarding the same case. In order to support education for this kind of discussions, we developed a discussion analysis tool called Corte with the following functions described below.

- Analysis of arguments in discussion records in terms of mathematical argumentation theories in order to determine which party succeeded in refuting the discussion, which argument becomes the key to conclude the discussion, and which argument needs to be discussed further.
- Calculate indices related to discussion skills using tags attached to each utterance in discussion record in order to evaluate:
  - "the composition of every argument" (whether the facts that should be claimed and legal theories were properly and clearly claimed or not), which is one of evaluation items in INC,
  - arbitration directions of mediators (whether they are facilitative or evaluative), which are considered to be the arbitration skills of mediators, and
  - the smoothness of arbitration (whether the discussion is rehashed or not), as another evaluation index for arbitration skills.
- Deadlock detection for supporting reframing analysis.

Corte inputs the discussion records, factor lists, and deadlock patterns.

Discussion records are text data where the utterance of participants are aligned in order of time. Factor lists are sets of factors that indicate the facts, laws, and claims that are included in the case to be analyzed. These lists are provided by the case maker or by discussion analyzers during the tagging

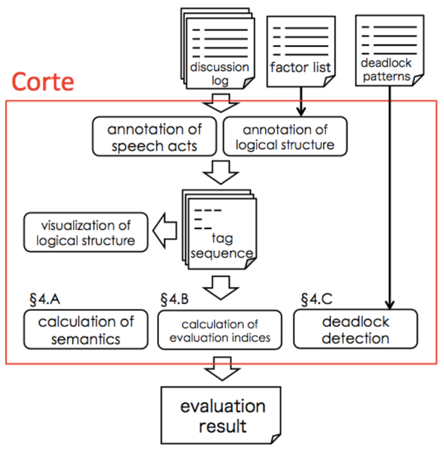


Figure 2. Workflow of discussion evaluation supported by Corte

process as described later. Deadlock patterns are chains of utterance types that indicate deadlocks to be extracted.

As shown by the flow in Figure 2, users of Corte annotate discussion records with speech act tags and argument structure tags. Based on this two-tag system, Corte evaluates discussions from various angles.

As speech act tags, in order to calculate evaluation indexes that will be described later and to detect candidates of deadlock portions, we used the following 11 categories: Claim, argument, agreement, denial, complement, close-ended-question (CEQ), open-ended-question (OEQ), reply, request, proposal, and "other".

Argument structure tags are argument elements where arguments included in discussions are dissolved based on the Toulmin model. However, the attack relationships between arguments are separately defined; the following three elements are sufficient when individual arguments are dissolved: Claim, warrant, and data. Each element corresponds to a factor on the factor list. Furthermore there are two types of relationships between elements, namely, support links and attack links. An argument consists of one or more elements and support links (a claim is supported by warrants and data). An element would be connected with other element via attack link since each element itself can be an argument. In the rest of this paper, we call this argument structure model as "argument model."

B. Annotation

Figure 3 shows the annotation screen of Corte. On the list of utterances shown in Figure 3, utterances are displayed using colors corresponding to each speaker. Factor lists show a list of argument elements based on the Toulmin model, while Section 3-A describes argument elements. Selecting each utterance shows its details on the top of the screen. In Figure 2 of "annotated factor", each user reads each utterance and adds elements on the diagram screen (in the center of the screen of Figure 3) while dividing utterances into argument structures. Doing this adds elements on the factor list shown on the right side of the screen. Annotating these added elements to the relevant portions can give the utterance log an argument structure tag. Although where there exist support or attack relationships between elements, the relationships between elements are defined by adding or deleting the arcs on the graph. With this tool, users can input argument structures just by editing the graph intuitively. At the same time, it can visualize argument structures (visualization of argument structures within Figure 2). On the screen shown by Figure 3, the elements included in the selected utterance (top of the screen) are indicated as nodes in thick color on the graph screen. In addition, between argument logs regarding a common case, sharing the argument lists and argument structure graphs can make it possible to compare discussions.

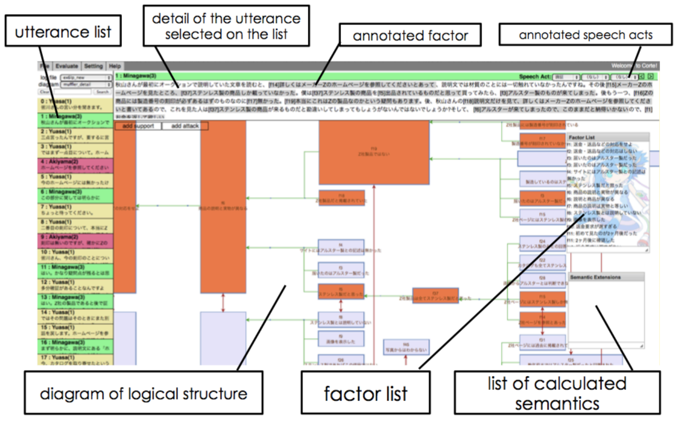

utterance list
detail of the utterance selected on the list
annotated factor
annotated speech acts
diagram of logical structure
factor list
list of calculated semantics

Figure 3. Screenshot of Corte

IV. Discussion Analysis by Using Corte

After annotation, this tool calculates argumentation semantics (section IV.A), calculates the evaluation indices for discussion skills (section IV.B), and detects deadlock portions (section IV.C). The section below explains the analysis methods based on the use of discussion logs of the actual mock arbitration and mock mediation, along with the results of evaluating the analysis methods. We used three mock arbitration logs and eight mock mediation logs for our experiments. Each log was evaluated by using the evaluation scales and evaluated by experts manually. We then compared the evaluation results. It should be noted that these educational-purpose discussions are rarely released to the public, while detailed evaluation conducted by experts are even less frequently publicized. Therefore, the number of logs to be evaluated is limited. For this reason, the section below describes the tendencies of evaluation based on the evaluation scale and evaluation conducted by experts. Note that we used discussion logs written in Japanese for that experiment, however, Corte supports Unicode files therefore other languages are also supported.

A. Semantic calculation

Semantic calculation is used to determine which argument can be established and not be established in a discussion where arguments and counter-evidence are entangled complicatedly. Here, while each argument structure for argument models is stored, semantic calculation is conducted by applying the argument model to the ASPIC+ framework [Prakken11] in order to generate the set of arguments and a set of attack relationships.

The argumentation system in the ASPIC+ framework can be defined by the following tuple

$$AS = (\mathcal{L}, \bar{\ }, \mathcal{R}, \leq) \tag{2}$$

where $\mathcal{L}$ indicates logical language, which is a set of inference elements, $\bar{\ } \subseteq \mathcal{L} \times \mathcal{L}$ indicates counter-evidence relationships between inference elements, $\mathcal{R} \subseteq \mathcal{L} \times \mathcal{P}(\mathcal{L}) \setminus \varnothing$ indicates inference regulations consisting of the strict inference regulations and the defeasible inference regulations, and $\leq$ indicates the partial order relationships in the inference regulations. In order to convert argument models in this paper into the argument system, we introduced the following conversion regulations.

1) Elements in the argument model are set to the logical language in the corresponding argument system.
2) Considering the warrant element to be one of the data elements, the claim elements to be the conclusion part and the data element to be the premise part, these elements are added to the defeasible rules in the corresponding argument system.
3) Attack relationships are added to the counter-evidence relationships of the argument system.

In the argument model, all elements are regarded as arguments, and there are possibilities that attack relationships could be generated between these elements. Therefore, all support relationships are interpreted to be defeasible rules. The order relationships in the inference regulations are considered to be always empty, because the order relationships of support relationships in the argument model are not considered. This simplifies the model. Figure 4 shows an example of converting the argument model into the argumentation framework by means of the procedure described above.

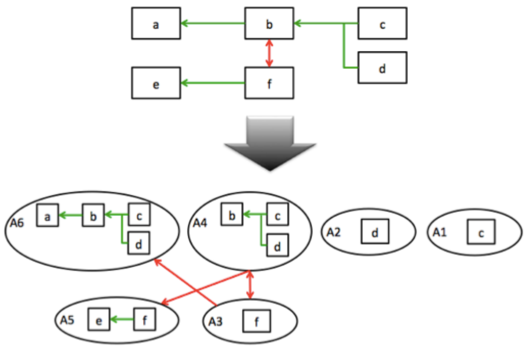


Figure 4. An example of extraction of argumentation framework from argument model

Figure 4 shows that the argument model which consists of six discussion points shown above changed into the argumentation framework which consists of six arguments shown below. After the argumentation framework has been obtained, this tool calculates each extension in the argumentation framework. In the case of Figure 4, preferred extensions are calculated to be A1, A2, A3, A5 and A1, A2, A4, A6. Each of these arguments originates in each discussion point of the argument model. Therefore, the list of discussion points corresponding to each semantic is presented on the list of extensions in the lower right of Figure 3. Figure 5 shows an example where arguments included in one of the preferred extensions as nodes emphasized in yellow. This figure overlaps argument graphs obtained from all discussion logs regarding the common case, while selecting and showing one of the preferred extensions in selected discussion logs. There are multiple preferred extensions. Therefore, this function is used to determine what argument is valid and what argument should be attacked so that the discussion can become advantageous.

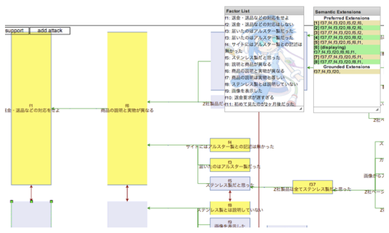


Figure 5. Nodes in the selected preferred extension is highlighted in yellow

B. Evaluation of discussion skills

Comparison and analysis of multiple discussions (match ups) based on common dispute cases can make it possible to clarify the issues for the participants in each discussion and evaluate discussions in a relative way. With that, introducing three indexes for evaluating discussion skills, we considered the effectiveness of these indexes.

1) Evaluation of the strictness (corresponds to the composition of every argument): The strictness is an index that indicates the composition of every discussion point (whether the facts that should be claimed and legal theories are properly and clearly claimed or not). Usually, the composition of issues is complicated in a negotiation competition, while multiple discussion points are included in one issue. According to each discussion point, the parties concerned extract discussion elements from the shared facts and previous knowledge, and present the arguments that attack the claims of the opponent party. By doing so, they need to increase the validity of their own claims. When discussions of both parties are engaged, arguments and counter-evidence arguments are repeatedly exchanged regarding the discussion point concerned, while the diagrams become quite dense. On the other hand, if discussions are not engaged, the claims of both parties remain superficial, while diagrams become non-dense. We obtained three discussion records (A, B, and C) on the first day (arbitration) of the INC in 2011. Figures 6 and 7 show the argument graphs that indicate the discussion points that were referred to during discussions A and C.

Figure 6. Factors referred on discussion A

These figures show diagrams of two discussions by overlapping them. The arguments appeared in discussion A are indicated in red nodes in Figure 6, while those appeared

Figure 7. Factors referred on discussion C

in discussion C are indicated in red nodes in Figure 7. Comparison of both figures clearly shows that Figure 6 has a greater number of discussion points than that of Figure 7.

In order to evaluate the strictness of this kind of discussions, we introduced the following scales.

- No. of support relationships: To what extent do the arguments justified by the warranty argument exist?
- No. of attack relationships: To what extent do the counter-evidence arguments against the opponent party's claim exist?
- Factor coverage (equals to No. of factors in the target discussion divided by No. of factors of all discussion logs): When other discussions are compared, to what extent is the argument elements presented?
- Warrant coverage (equals to No. of warranty elements that appear in the target discussion divided by No. of warranty elements that appear in all discussion logs): When other discussions are compared, to what extent are the legal regulations including contracts in operation?

TABLE I. STRICTNESS CORRESPOND TO EACH DISCUSSION

index	*A*	*B*	*C*
support relationships	55	39	31
attack relationships	28	18	18
Factor coverage	93	59	52
Warrant coverage	79	36	36

Table I shows the calculation results of the evaluation scales in the third discussion of the INC of 2011. This shows that as these values became greater, as the discussion proceeded strictly and logically. Table 1 shows the order of the strictness according to the discussion as A≪B≤C. This shows that discussion A was stricter than discussions B or C.According to the evaluation results of INC, team X participated in discussion A received the top prize. The other team that participated in discussion B was ranked lower than team X, but was awarded the top prize. Additionally, team Z participated in discussion C but never received the top prize. The evaluation done by legal experts shows that discussion A was highly evaluated in the item "the composition of every discussion point," while there was no significant difference between discussion B and discussion C. This result shows that the evaluation values of the strictness of discussion performed by the experts showed similar tendencies in the evaluation of "the composition of every discussion point." This suggests that strictness can be used for evaluating discussion skills.

2) Evaluation for mediator's attitude: The attitude shown by mediators during an arbitration case can be classified into facilitative or evaluative. Being facilitative is an attitude that facilitates the parties concerned to make utterances while showing respect for the utterances made by the participants concerned. Being evaluative is an attitude that evaluates the utterances made by the parties concerned while the discussions proceeds using leadership. Generally, mediators are required to be facilitative. In other words, even though mediators have more expertise than the client, they are prohibited to give advantageous information to either side. They have to protect their neutral position.

It is necessary for mediators to show facilitative attitude in an arbitration case. In a mock arbitration, however, many students play the roles of mediators. Therefore, they need to have proper instructions. Determination of mediators' attitude from arbitration records can be used for reference to teach arbitration skills to mediators.

Differences in attitudes are determined focusing on the speech act tag, proposal. Facilitative mediators behave so that the parties concerned can make positive utterances. However, evaluative mediators present their own evaluation, or sometimes they even take the lead in forming consensus. On the other hand, in the process of forming consensus, an utterance with the intention to form a proposal appears. For this reason, the attitude of mediators is evaluated by using the percentage of the number of proposal utterances made by the mediators. Namely, this percentage is expressed by No. of proposal utterances made by the mediators divided by No. of all utterances in each discussion log. Although the arbitration issue is the same, the number of utterances may vary significantly. Therefore, the proposal utterances are divided by all utterances in order to reduce this influence by normalization.

TABLE II. MEDIATOR'S ATTITUDES ON EACH DISCUSSION

index	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*
Proposals by mediator (1)	0	3	1	0	3	2	0	5
Total utterances (2)	76	68	91	25	70	55	67	62
(1) / (2) [%]	0	4.4	1.1	0	4.3	3.6	0	8.0
expert's evaluation	F	E	F	F	E	F	E	E

By using eight discussion records of mock arbitration cases regarding products between sellers and bidders for buying and selling automobile mufflers in an online auction, we verified the evaluation scale that indicates whether the mediators were evaluative or facilitative. Table II shows the calculation results of evaluation scale and the evaluation results of each discussion done by legal experts. According to Table II, this evaluation index value shows the same tendency of the evaluation done by experts. It also suggests the possibility of automatic determination by the mediators' attitude.

3) Evaluation for smooth operation of arbitration: One more index for measuring the skills of mediators is whether the arbitration discussion proceeds smoothly or not. In arbitration, after the parties concerned have confirmed the facts and their desires, each party proposes solutions based on the facts recognized by them, in order to reach consensus.

A certain order where consensus is formed after the facts have been confirmed is very important for smooth consensus formation. When facts are reconfirmed during the discussion of consensus formation, consensus items formed during the past consensus formation processes could be invalid. As to the scale for measuring whether the discussion goes back to fact confirmation after the consensus formation discussion has started, the value normalized by the number of proposal utterances after the first proposal utterance has been made with the number of argument elements presented after the first proposal utterance has been made. Namely, as the smooth operation level of each discussion log, No. of factors referred to after the first proposal utterance has been made divided by No. of utterances made after the first proposal utterance has been made is introduced. Here, the discussion of consensus formation is interpreted as to have started by the first proposal utterance.

TABLE III. SMOOTHNESS INDICES ON EACH DISCUSSION

index	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*
Factors after the first prop. (1)	0	15	0	0	0	3	4	0
Utterances after the first prop. (2)	32	53	68	8	24	32	28	28
(1) / (2) [%]	0	28	0	0	0	9	14	0
expert's evaluation	A	C	B	B	A	C	C	B

Table III shows the calculation results of evaluation scales regarding the smooth operation level of mediators by using the same logs to evaluative attitude and the verification results by using the evaluation results of each discussion done by experts. The experts' evaluation show that evaluation A (excellent) was smoother than evaluation B (good) and C (poor).

According to Table III, the experts determined that discussions with evaluation indexes more than 0 are not smooth. However, distinguishing between good and excellent was not determined. However, discussions which were not extremely smooth corresponded to the evaluation done by experts.

C. Deadlock detection

Discussions could go into deadlock when both parties do not yield. In order to eliminate deadlocks, reframing utterances to change viewpoints are effective. Deadlocks have a wide variety of patterns; it is difficult to enumerate these patterns.

Corte users register deadlock patterns, and Corte detect portions that match with the deadlock patterns. By doing so, this tool offers deadlock recognition support. The following pattern is a sample pattern. A typical deadlock pattern is that a proposal made by one party is rejected by the other party without having any particular reasons. This phenomenon is repeated again and again. In other words, where the parties concerned (speakers) are A and B, the factors concerned are X and Y, and the following tuples (speaker, speech act tags, and logical structure tags) are used, the utterance systems below are repeated.

1) (A, proposal, X) OR (A, proposal+argument, X)
2) (B, denial, $\neg$X) OR (B, denial+argument, Y$\Rightarrow$ $\neg$X)

TABLE IV. PERFORMANCE OF DEADLOCK DETECTION

Index	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*Total*
Detected by Corte	3	3	1	1	2	2	3	0	15
Detected manually	1	0	2	0	1	1	3	2	10
Precision	1/3	0/3	0/1	0/1	1/2	1/2	1/3	0/0	4/15
Recall	1/1	0/0	0/2	0/0	1/1	1/1	1/3	0/2	4/10

Table IV shows the results of deadlock detection from discussion logs of mock arbitrations by applying deadlock patterns. While this tool successfully detect a part of deadlocks, its accuracy is not enough high. Therefore, it is clear that additional patterns or features are necessary for detection.

V. CONCLUSION AND FUTURE WORK

In this paper, we introduced objective evaluation methods to evaluate the entire discussion by giving logical structure tags and speech act tags to discussion utterance logs, methods to analyze and detect deadlock portions, and the discussion evaluation support tool Corte. We also evaluated this proposed tool by using the actual mock arbitration and mock mediation logs. Our evaluation results showed that the tool output showed a similar tendency to the evaluation results obtained by experts. On the other hand, when it comes to detecting deadlock portions, detection by using tags is partially effective, while deadlock patterns need to be analyzed more.

Future issues include the enhancement of analysis and evaluation discussion logs. Annotation was done manually in the tool proposed this time. We are actually working on semi-automatic annotation [15]. Automatic extraction of tag patterns of deadlock portions needs to achieve further energy-saving measures for users.

REFERENCES

[1] Intercollegiate Negotiation Competition. http://www.osipp.osaka-u.ac.jp/inc/index.html.

[2] Toulmin, S. 1958. The Uses of Argument, Cambridge University Press.

[3] Reed, C. and Rowe, G. 2001. Araucaria: Software for Puzzles in Argument Diagramming and XML. Technical Report. Dept. of Applied Computing. University of Dundee.

[4] Nitta, K. et. al. 2002. A Legal Negotiatiton Support System Based on A Diagram (in Japanese), Transactions of the Japanese Society for Artificial Intelligence, Vol.17, No.1 D, pp.32-43.

[5] Shibata, Y. and Yamaguchi, K. 2011. SPURI: Argument Analysis Framework (in Japanese), Journal of Information Processing, 52(3), pp. 1395-1411.

[6] Lynch, C., Ashley, K. D., and Falakmassir, M. 2012. Comparing Argument Diagrams. In JURIX 2012: The Twenty-Fifth Annual Conference on Legal Knowledge and Information Systems. pp. 81-90.

[7] Hashida, K. 1998. GDA Versatile and Intelligent Contentware with Semantic Annotation (in Japanese). Transactions of the Japanese Society for Artificial Intelligence, Vol.13, No.4, pp.528-535.

[8] Tanaka, T. 2007. Case Based Online Mediation Support System (in Japanese). Doctoral Thesis. Tokyo Institute of Technology.

[9] Dung, P. M. 1995. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. Artificial Intelligence, 77, (2): pp. 321-357.

[10] Prakken, H. 2011. An overview of formal models of ar- gumentation and their application in philosophy. Studies in logic. Vol 4, no 1: pp. 65-86.

[11] Ashley, K. 1991. Modeling legal arguments: Reasoning with cases and hypotheticals. MIT Press.

[12] Aleven, V. 1997. Teaching case based argumentation through an example and models. Doctoral Thesis. The University of Pittsburgh.

[13] Levin, K., H. 2007. Chouteisya Handobukku - Choutei No Rinen To Gihou (in Japanese). Shinzansya.

[14] Hall, L., M. and Bodenhamer, B., G. 2011. MIND-LINES: Lines For Changing Minds, 5th ed (Yuile, Y. trans.). Empowerment Technologies (Original work published in 1997).

[15] Kubosawa, S., Lu, Y., Okada, S. and Nitta, K. 2012. Argument Analysis with Factor Annotation Tool. In JURIX 2012: The Twenty-Fifth Annual Conference on Legal Knowledge and In- formation Systems. 61-70.

Program Optimization Strategies to Improve the Performance of SpMV-Operations

Rudolf Berrendorf*, Max Weierstall*, Florian Mannuss†
* Computer Science Department, Bonn-Rhein-Sieg University, Sankt Augustin, Germany
e-mail: rudolf.berrendorf@h-brs.de, max.weierstall@h-brs.de
† EXPEC Advanced Research Center, Saudi Arabian Oil Company, Dhahran, Saudi Arabia
e-mail: florian.mannuss@aramco.com

***Abstract*—The SpMV operation – the multiplication of a sparse matrix with a dense vector – is used in many simulations in natural and engineering sciences as a computational kernel. This kernel is quite performance critical as it is used, e.g., in a linear solver many times in a simulation run. Such performance critical kernels of a program may be optimized on certain levels, ranging from using a rather coarse grained and comfortable single compiler optimization switch down to utilizing architecural features by explicitly using special instructions on an assembler level. This paper discusses a selection of such program optimization techniques in this spectrum applied to the SpMV operation. The achievable performance gain as well as the additional programming effort are discussed. It is shown that low effort optimizations can improve the performance of the SpMV operation compared to a basic implementation. But further than that, more complex low level optimizations have a higher impact on the performance, although changing the original program and the readability / maintainability of a program significantly.**

Keywords–Sparse Matrix Vector multiply (SpMV); Single Instruction Multiple Data (SIMD); OpenMP, unrolling; intrinsics.

I. INTRODUCTION

Sparse matrices are widely used in many areas of natural and engineering sciences [1], escpecially in simulations. An often used operation on such matrices is the multiplication of a sparse matrix with a dense vector (SpMV). This operation is often the most time consuming operation in iterative solvers (e.g., CG, GMRES [1]), which are the most time consuming operations in many simulations. Therefore, much attention has been given to optimize this operation. One point in an optimization discussion is the choice of an appropriate storage format for the sparse matrix [2]–[6], which depends mainly on the given matrix stucture (e.g., high / low matrix bandwidth, etc.) and the target architecture, e.g., multicore CPU, multiprocessor system, Graphics Processor Unit (GPU). Compressed Sparse Row (CSR) [1] is a general storage format for sparse matrices that performs quite good, especially on CPU-based systems. But even for a fixed storage format like CSR, there are opportunities for program optimization on different abstraction levels.

CPUs and memory systems are optimized for specific workloads in programs. Other than utilizing the memory hierarchy, instruction pipelining and vector units in processors can have a significant influence on a program's performance. For instruction pipelining, large basic blocks are favorable in a program. All recent processors have also some implementation of vector registers and related vector instructions [7] that can significantly speed up computations that exploit this architectural feature. Compilers can optimize code with large basic blocks with much room for optimizations and by vectorizing loops [8]–[12], as long as all data dependencies are respected [13].

Some high level programming models, especially designed for parallel (and therefore resource intensive) computing, have some notations to give hints to a compiler concerning vector operations. For example, OpenMP [14] as the de facto standard for shared memory parallelism has got in the recent revision 4 some annotations to guide a compiler in using vector units in a processor. But vectorizing compilers and directives to give a compiler hints existed already before OpenMP [15].

Other than leaving every optimization to a compiler, there are program optimization techniques known that allow to restucture a program to optimize certain operations (that a compiler may not detect). This restructuring of source code can be done by an expert programmer or by a sophisticated tool [16] [17] [10].

And, in a third way, if for example a compiler is not able to generate fast code because for example complex index expressions exist, a programmer may use vector intrinsics on a rather low abstraction level to program the hardware directly on a more or less assembler level [18]. For some high level language like C / C++, this can be accomplished by using compiler extensions called intrinsics that look like functions calls in the programming language and correspond to one or few assembler instructions.

In a summary, there are certain levels on which a time consuming operation may be speed up. In this paper, the question will be answered for the SpMV operation what the programming effort is needed for an optimization and what performance improvement one can get, if any.

The paper is structured as follows. Section II gives an overview on related work. After that, Section III discusses some program optimization techniques that are used for the investigations in this paper. Section IV describes the test environment for our evaluation. Section V shows and discusses performance results. The paper is summarized in Section VI.

II. RELATED WORK

Compiler writers give hints in user's guides [8] and technical notes [19] how to optimize programs and how to write programs in a way such that a compiler can apply optimizations. Further than that, there exists optimization guides with a detailed description of hardware features that a programmer can use [20] [21].

Dedicated to the SpMV operation many papers were published describing performance related program optimizations of various types (e.g., register blocking) that were applied in [22]–[25].

In [26], Wende discusses the use of SIMD functions (i.e., vector intrinsics) to impove the performance on Intel Xeon processors and Xeon Phi coprocessors [27] escpecially for branching and conditional functions calls. He found that for this special application scenario there are only rare situations with a performance improvement by using vector intrinsics. This was mostly the case if the ratio of arithmetic operations to control logic is low.

III. Program Optimizations

Nowadays, processor and memory architecures are rather complex. Many architectural optimitations have been done in last decade's processor architectures that may improve the performance of programs significantly. Such architectural improvements include multi scalarity, out-of-order execution, pipelining and many others [7]. For all enumerated hardware optimizations it is favorable to have large basic blocks (code without any branches). Unfortunately, the SpMV has a rather small loop body for many storage formats. A well-known technique called loop unrolling [10] [11] enlarges the basic block of a loop body.

A significant performance boost for many applications is the use of vector registers / units that are available in almost all recent processor architectures [28]–[30]. These vector architectures follow the well-known SIMD principle [31] that one instruction is applied to several operands at the same time. For a vector width of n data elements, this may result in a speedup of up to n. Recent processors eligible in High Performance Computing (HPC) have a vector register / unit width of up to 256 bits, corresponding 4 double precision elements, each 64 bits. Recent announcements show [32] that the vector width will double in the near future with a nominal floating point performance increase of a factor of two.

The enlargement of basic blocks in a loop body and the use of vector registers can be used to speed up an SpMV operation. There are now certain levels of abstraction on which a programmer may influence these (and other) optimizations.

A. Compiler Flags

A simple strategy is to leave any optimization to a compiler. A programmer may specify on a rather coarse level some general compiler optimization level (i.e., `-O2, -O3`) leaving any decisions and optimization strategies solely to the compiler according to the specified optimization level. Specifying an optimization level of 2 instructs many compilers to enable many optimization techniques that have no influence on the semantics of a program, i.e., no optimizations are applied that may change the meaning of a program as for example using a faster floating point artithmetic.

Additionally on a finer level, special compiler options can be used to include certain optimization techniques or to utilize certain architectural features. An example for that is to allow the generation of code that utilizes the latest additions in the instruction set. For example, the compiler option `-march=haswell` of the GNU compiler g++ [33] allows the generation of advanced instructions only available on Haswell processors. Alternatives would be for the previous generations of Intel processors `-march=ivybridge` or `-march=sandybridge`. The code may be no longer executable on processors of generations previous to the one specified. Other compilers have the same possibilities but with a different syntax of such an option. Without the specification of such an architectural option the compiler generates code with an instruction set that corresponds by default to rather old processors to allow the compiled program to run on many systems, even older ones.

The PGI compiler [34] offers an option to instruct the compiler to generate vector code utilizing vectors of a specific size. For example the option `-tp=haswell -Mvect=simd:256` directs the compiler to generate code for Haswell processor, i.e., utilizing the Advanced Vector Extensions 2 (AVX2) instruction extensions, and to work with vectors of up to 256 bits.

```
#pragma omp simd reduction(+:s)
for(int i==; i<n; i++)
    s += a[i];
```

Figure 1. Example code for the use of an OpenMP pragma.

B. Language Directives

Sometimes, a compiler may not be able to recognize that certain optimization techniques could be applied to a code sequence. For example, this may be the case because the compiler can not know at compile time the value of certain variables, the alignment of variables or cannot exclude data dependencies because of complex index expressions. But, if a programmer can assure that for example a certain variable is always larger than 100 the compiler could optimize this program code. There exist program annotations for exactly these situations to tell a compiler some additional semantic information. Dependend on the programming language this may be done in different ways.

An example is OpenMP [14] where in the fourth version of this standard certain extensions were added that allow a programmer to specify (among parallelism, which is the main focus of OpenMP) that certain parts of a program should be vectorized by the compiler, including hints how to do that or assumptions that a compiler can rely on at that point of the program.

A small example for that is the piece of code shown in Figure 1. Here, the pragma tells the compiler to vectorize the loop and to handle the variable `s` as a reduction variable with a special treatement (this is necessary due to the loop carried data dependence on `s`).

The `simd` directive requests to vectorize that part of a program that is in the scope of this directive. For the `simd` directive there are additional clauses beside the shown **`reduction`** clause possible mainly assuring certain program properties. Among them are:

- `aligned` specifies that the specified data objects are aligned to a certain byte boundary.

```
double haddSum( __m256d tmp) {
    // vecA := ( x2 , x1 )
    const __m128d vecA = _mm256_castpd256_pd128(tmp);
    // vecB := ( x4 , x3 )
    const __m128d vecB = _mm256_extractf128_pd(tmp,1);
    // vecC := ( x4+x3 , x2+x1 )
    const __m128d vecC = _mm_hadd_pd(vecA,vecB);
    // vecS := ( x4+x3+x2+x1 , x4+x3+x2+x1)
    const __m128d vecS = _mm_hadd_pd(vecC,vecC);
    // returns x4+x3+x2+x1 as double
    return mm_cvtsd_f64(vecS);
}
```

Figure 2. Example code for the use of compiler intrinsics.

- `safelen` guaranties that n consecutive iterations can be executed in parallel / are independent.
- `linear` tells the compiler that the loop variable has a linear increase.

Similar compiler directives `simd` (vectorize code) and `ivdep` (ignore vector dependencies) outside of the OpenMP standard are known to several compilers with a similar meaning. We have seen no large differences in results for these alternatives.

An (non-OpenMP) directive that many compilers recognize in one or the other notation is the hint to unroll a loop [10]. This can be favorable if the loop body is rather small (as with the SpMV operation) and therefore the instruction pipeline runs soon out of instructions. Additionally, with a larger basic block a compiler may have more opportunities to optimize, e.g., to keep reused values in registers.

C. Vector Intrinsics

A compiler needs to generate special vector instructions to utilize the vector units in a processor. Sometimes a compiler may not be able to detect an appropriate situation because the data dependence analysis in a compiler cannot safely exclude any dependencies. Or a compiler generates sub-optimal code for that situation. In such situations, a programmer may himself “generate“ vector instructions by using so called vector intrinsics.

Vector intrinsics [18] are available with some widely used compilers, e.g., GNU g++ [33], Intel compiler icpc [8]. With these intrinsics a programmer has more or less direct access to vector instructions of the underlying hardware. But please be aware that this functionality is on the level of assembler intructions where one has to manage vector registers and vector instructions directly.

The example in Figure 2 shows how to add 4 values using vector intrinsics. `__m128d` and `__256d` are special vector types and `_mm...` are function calls that correspond to vector instructions. This small example makes it very clear that using intrinsic functionality makes a program hard to read / understand because hardware features are programmed embedded within a high level language like C or C++.

D. How to Choose the Right Program Optimization Strategy?

This spectrum of optimization techniques shown above has consequences for programmers. The first approach (use a compiler switch) leaves any decision and optimization to the compiler. This is a possibility that is quite comfortable for a programmer and does not require any sophisticated skills from a programmer.

The next possibility is to leave many things to the compiler but to give additional hints to the compiler using pragmas / directives. A compiler bases its decision concerning vectorizability (and many other optimizations) on data dependency information [13]. When a compiler cannot decide if a part of a program is optimiziable / vectorizable, the opportunities that a hardware architecture gives to dispose cannot be utilized. But with this approach a programmer needs experience and expert knowledge how a compiler works and what information it may miss in certain program parts. If a programmer assures wrong properties (e.g., safe distance of iterations) a compiler may even generate wrong code. If a programmer uses such directives the programming effort (additional lines of code) is rather small.

The last option is to allow a programmmer direct access to the functionality a hardware provides. This allows to utilize the available funcionality in an efficient way. Performance-aware programmers are used to such things. But this has severe consequences. One point is that the programming level is quite low and the resulting program is therefore hard to write and read. Additionally, programming is now getting platform specific, i.e., a program kernel developed and optimized for an Intel Haswell system is not executable on / not optimized for an older Ivy Bridge / Sandy Bridge system. This means, that any company using such advanced features has to provide an expert that is aware of all technological feature of hardware generations in use and how to use them.

Comfortability to the programmer is one aspect of consideration. If this would be the only aspect it is clear that the approach would be used to leave everything to the compiler. Many simulations in natural science run for hours, days or even weeks. Often a large part of the runtime is executed in rather small parts of the program, computationally intense program kernels like the above mentioned SpMV operation. For such really performance critical parts of a program all possibilities are analyzed that may lead to a decrease in runtime, even on the intrinsic level.

Therefore, the question is whether and if yes how much can a program benefit from programming techniques? Or is an optimizing compiler able to deliver the same (or even better) performance?

IV. Experimental Setup

To answer these questions we used the (rather small) compute intensive SpMV program kernel. We used our own implementation of the SpMV operation in C/C++ using the CSR storage format [1]. The basic implementation we use in our subsequent comparism is shown in Figure 3 (here shown in a rather simplified and compact version).

We used four different versions (as an information in parentheses the number of lines of code to realize that):

- *normal*: the unmodified version similar to the above shown (7 lines)
- *unroll*: the compiler was told with a directive to unroll the loop four times (8 lines)
- *simd*: the compiler was told with a directive to vectorize the code / to generate vector instructions (8 lines)

```
void SparseMatrixCSR::SpMV(Vector &v, Vector &u) {
  // iterate over all rows of the matrix
  for(int i=0; i<nRows; i++)
    // handle all non-zero elements in a row
    for(int j=rowStart[i]; j<rowStart[i+1]; j++)
      u[i] += values[j] * v[columnIndex[j]];
}
```

Figure 3. Basic code version for the SpMV operation (simplified).

TABLE I. SYSTEMS USED.

system name	SB	Haswell
instruction set	AVX	AVX2
architecture	Sandy Bridge EP	Haswell EP
processor (Intel Xeon)	E5-2670	E5-2680 v3
cyle time in GHz (TurboBoost)	2,6	2,5

- *intrinsics*: our own implementation using vector intrinsics (97 lines). The code contains a distinction, which vector instruction set AVX or AVX2 should be used and different intrinsics must be used in some parts of the program, dependent on the instruction set.

To measure performance numbers we use systems of different generations of Intel processors (Intel Sandy Bridge and Haswell). Table I gives an overview of relevant system parameters and systems names. The older Sandy Bridge generation supports only the AVX instruction set, in newer Haswell systems additional features are available in the AVX2 instruction set.

We used matrices as data sets with different properties that may influence performance, e.g., the distribution of non-zero values in a row. In total, 110 matrices were used. The matrices are taken form the Florida Sparse Matrix collection [35] and from the Society of Petroleum Engineers (SPE) challenge [36].

We used two compilers in recent versions:

- *g++*: GNU g++ vesion 5.2.0 [33]
- *icpc*: Intel icpc version 15.0.1 [8]

To filter accidental effects that may happen on any system each measurement was repeated 100 times and the median was taken as the measurement value.

V. Evaluation

We discuss each optimization independently and summarize with an overall comparism. We give statistical values over all 110 matrices and additionally give a single absolute value for the SPE matrix `spe5Ref_a`, which shows often a similar behaviour compared to many other matrices and is used therefore as a representative.

A. Influence of Compilers and Compiler Levels

Both compilers in use provide compiler switches to turn on certain global optimization levels: `-O0` up to `-O3`. The optimization level 0 should be used for debugging only and not for production runs. The Intel compiler provides further an additional level `-fast` and the GNU compiler the option `-Ofast` to additionally turn on processor specific optimizations as well as interprocedural optimizations for the Intel compiler. But with this option the code eventually runs no longer on processors of previous generations while with the option `-O3` the code is still runnable on all recent systems. The default value for g++ is no optimization, the default for icpc is level 2.

TABLE II. RUNTIMES IN MILLISECONDS FOR VARIOUS COMPILER OPTIMIZATION LEVELS FOR THE EXAMPLE MATRIX `spe5Ref_a`.

	g++		icpc	
optimization option	SB	Haswell	SB	Haswell
-O0	342	295	379	308
-O1	173	139	150	139
-O2	96	87	150	137
-O3	93	82	150	136
-(O)fast	93	82	139	83

TABLE III. EFFECT OF UNROLLING (SEE TEXT FOR EXPLANATION OF ITEMS).

	g++		icpc	
	SB	Haswell	SB	Haswell
% instances better	80	72	22	14
runtime exa. matrix [ms]	90	81	153	162
minimum speedup	0.736	0.790	0.548	0.723
maximum speedup	1.129	1.700	1.293	1.111
average speedup	1.011	1.027	0.967	0.966
standard deviation	0.061	0.101	0.081	0.052

Table II shows as a representative example the results for the matrix `spe5Ref_a` on the Sandy Bridge and Haswell system. The results are transferable to the other matrices and are therefore general statements concerning our SpMV implementation. The performance of the GNU compiler generated code increases with each level while the performance of the Intel compiler is more or less good and nearly the same for all levels other than 0. The option `-fast` with the Intel compiler produces (processor specific) code that is significantly faster than the code of the other optimization levels. It should be noted that these results are specific to this program kernel and results may be different for other program kernels dependend on the source code and for other/future compiler versions.

As the compiler run itself does not take any significant amount of time for any optimization level it is recommandable always to invoke the compiler with a high optimization level. As there are no semantic problems with our code with level 3, we use this level in all following discussions and name it the base case.

B. Unrolling Loops

The next optimization technique we looked at is loop unrolling to enlarge basic blocks. This should optimize register usage and reduce the loop overhead for small loop bodies (as in our case). As already discussed, this can be used rather comfortable with directives specifying before a loop that this loop should be unrolled and giving an unroll factor. We found out empirically that an unroll factor of 4 performed best.

Table III shows results for using unrolling. The first line (% instances better) shows how many of the 110 matrices in percent showed a better performance result using this technique compared to the base case (with the same compiler). The next row (runtime exa. matrix) gives the runtime for the example matrix `spe5Ref_a` in milliseconds to be able to compare results directly. Minimum and maximum give the minimum / maximum speedup value compared to the base value. For example, a value of 0.736 for the minimum speedup means that the worst problem instance shows only 73.6 percent

TABLE IV. EFFECT OF `simd` DIRECTIVES (SEE TEXT ON UNROLLING FOR EXPLANATIONS OF ITEMS).

	g++		icpc	
	SB	Haswell	SB	Haswell
% instances better	1	0	56	73
runtime ex. matrix [ms]	147	158	148	145
minimum speedup	0.584	0.486	0.532	0.535
maximum speedup	1.076	1.000	1.121	1.326
average speedup	0.698	0.602	0.969	1,034
standard deviation	0.095	0.100	0.100	0.147

TABLE V. EFFECT OF INTRINSICS (SEE TEXT ON UNROLLING FOR EXPLANATIONS OF ITEMS).

	g++		icpc	
	SB	Haswell	SB	Haswell
% instances better	93	92	86	91
runtime ex. matrix [ms]	77	66	97	93
minimum speedup	0.805	0.631	0.565	0.622
maximum speedup	1.385	1.545	1.252	1.525
average speedup	1.153	1.184	1.093	1.237
standard deviation	0.107	0.145	0.128	1.196

of the performance of the base case. Average and standard deviation are the average speedup and standard deviation of the speedup over all 110 problem instances. An average of 1 and more means that the used technique performed in average better that the base case.

The results show that for both compilers the performance effect is more or less neglectable. The GNU compiler shows in average a minimal improvement while the Intel compiler shows instead a minimal performance degradation. This can be explained because the compilers themself apply already optimization techniques that make an explicit unrolling unfavorable.

C. *Using Vector Directives*

Table IV shows results for the use of OpenMP `simd` directives to explicitly request a vectorization. Enabling OpenMP vectorization with the GNU compiler results in a significant performance loss for nearly all problem instances. This may be attributed to the fact that the support for this OpenMP functionality is rather new in the compiler and causes rather performance-worsening code generation compared to the (good) optimization with the compiler level `-O3` of the base case. The Intel compiler shows in average rather similar results compared to the base case.

D. *Using Intrinsics*

Table V shows results for the use of (processor specific) intrinsics. The best absolute results were achieved for most matrices with this version. Only with few test matrices a small performance degradation could be seen as the matrix structure could not take advantage out of vector processing. For the Intel compiler there is a significant boost in performance as well. But here it is fair to say that the Intel compiler option`-fast` generates also processor specific code like with the intrinsics and is even faster.

E. *Summary*

This section summarizes the results. Table VI summarizes the absolute run times for the various optimizations on the example matrix. Figure 4 shows the percentage of problem

TABLE VI. SUMMARY OF RUN TIMES IN MILLISECONDS FOR THE EXAMPLE MATRIX.

	g++		icpc	
	SB	Haswell	SB	Haswell
-O0	342	295	379	308
-O1	173	139	150	139
-O2	96	87	150	137
-O3	93	82	150	136
-(O)fast	93	82	139	83
unrolling	90	81	153	162
simd	147	158	148	145
intrinsics	77	66	97	93

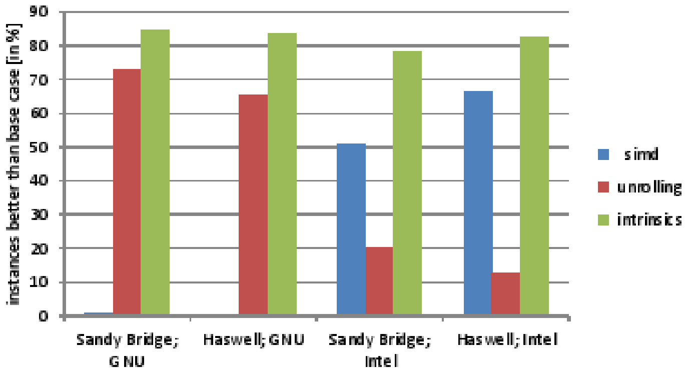


Figure 4. Percentage of instances that performed better than the base case.

instances that got a performance gain when applying an optimization technique.

The first technique used was compiler flags. The additional effort and needed knowledge for using compiler flags is minimal and no code change is necessary. The results were twofold: The Intel compiler uses already optimization techniques with lower optimization levels and has not shown any further improvements with higher optimization levels. Only the option `-fast` (producing processor specific code) resulted in an additional performance boost. For the GNU compiler, there was a steady performance increase with higher optimization levels.

The unrolling technique did not show any major change in runtime. Here, the compilers did already a good job. The programming effort and necessary expertise to use it is quite low.

Using the `simd` compiler directive to ask for vectorization is easy to use but requires a deeper knowledge, e.g., on data dependencies to avoid wrong code generation. The performance reached with the GNU compiler on the beside the directive unmodified code was worse compared to the base case. Using the Intel compiler this directive has no significant influence.

To use intrinsics a deep understanding of a processor architecture and the available instruction set is necessary. Additionally the algorithm may be quite different to the normal version when expressing it on an intrinsic level. The program code is totally different to the original code and quite hard to read and write, at least for a programmer who is not used to intrinsics. But the overall best performance results were reached with intrinsics.

In a nutshell, all optimizations other than `simd` directives with the GNU compiler and loop unrolling with the Intel compiler had a positive influence for most problem instances. Using intrinsics resulted in 80% and more problem instances for a performance improvement.

VI. Conclusions

SpMV is a time critical operation in many applications. Optimizing this operation is a challenge. There are various opportunities to tackle that problem on a program optimization level, partially dependend on the compiler used.

In this paper, several optimization approaches were described and compared to each other concerning programming effort / required expert knowledge and achieved performance. It was shown that using a simple compiler switch for the highest level of optimization a compiler can often get near to 80% of the best performance reached with any other approach. Unrolling is easy to use but did not show any significant performance effect. Using OpenMP `simd` directives showed only a small performance impact, although some problem instances gained more performance. Using this directives with the GNU compiler is currently not encouraged. Rewriting the program kernel with vector intrinsics means a total rewrite of the code, which is only acceptable for really compute intensive and rather small program kernels. But the performance gain can be substantially increased and this approach resulted in the best performance for the SpMV operation.

Acknowledgements

Jan Ecker, Javed Razzaq and Simon Scholl at Bonn-Rhein-Sieg University helped us in many discussions. We would also like to thank the CMT team at Saudi Aramco EXPEC ARC for their support and input. Especially we want to thank Ali H. Dogru for making this research project possible. We would like to thank the anonymous reviewers for their suggestions and comments.

References

[1] Y. Saad, Iterative Methods for Sparse Linear Systems, 2nd ed. SIAM, 2003.

[2] M. Kreutzer, G. Hager, G. Wellein, H. Fehske, and A. R. Bishop, "A unified sparse matrix data format for efficient general sparse matrix-vector multiply on modern processors with wide SIMD units," SIAM Journal on Scientific Computing, vol. 26, no. 5, 2014, pp. C401–423.

[3] W. Liu and B. Vinter, "CSR5: An efficient storage format for cross-platform sparse matrix-vector multiplication," in Proc. 29th Intl. Conference on Supercomputing (ICS'15). ACM, 2015, pp. 339–350.

[4] K. Li, W. Yang, and K. Li, "Performance analysis and optimization for SpMV on GPU using probalistic modeling," IEEE Trans. Parallel and Distributed Systems, vol. 26, no. 1, Jan. 2015, pp. 196–205.

[5] W. Liu and B. Vinter, "Speculative segmented sum for sparse matrix-vector multiplication on heterogeneous processors," preprint arXiv:1504.06474v1, 2015.

[6] J. Wong, E. Kuhl, and E. Darve, "A new sparse matrix vector multiplication GPU algorithm designed for finite element problems," Intl. Journal for Numerical in Engineering, Jan. 2015, pp. 1–35.

[7] J. L. Hennessy and D. A. Patterson, Computer Architecture: A Quantitative Approach, 5th ed. Morgan Kaufmann Publishers, Inc., 2012.

[8] User and Reference Guide for the Intel C++ Compiler 15.0, https://software.intel.com/en-us/compiler_15.0_ug_c ed., Intel Corporation, 2014, retrieved: February 2016.

[9] LLVM Project, The LLVM Compiler Infrastructure, http://llvm.org/, retrieved: February 2016.

[10] R. Allen and K. Kennedy, Optimizing Compilers for Modern Architectures. San Francisco: Morgan Kaufmann, 2002.

[11] K. D. Cooper and L. Torczon, Engineering a Compiler, 2nd ed. Burlington, MA: Morgan Kaufmann, 2012.

[12] S. S. Muchnick, Advanced Compiler Design and Implementation. San Francisco: Morgan Kaufmann, 1997.

[13] U. Banerjee, "An introduction to a formal theory of dependence analysis," The Journal of Supercomputing, vol. 2, 1988, pp. 133–149.

[14] OpenMP Application Program Interface, 4th ed., OpenMP Architecture Review Board, http://www.openmp.org/, Nov. 2015, retrieved: February 2016.

[15] Cray Research Inc., CF90 Commands and Directives Reference Manual, 1995, sR-3901 2.0.

[16] R. Eigenmann, J. Hoeflinger, G. Jaxon, Z. Li, and D. Padua, "Restructuring Fortran programs for Cedar," Concurrency - Practice and Experience, vol. 5, no. 7, Oct. 1993, pp. 553–573.

[17] K. A. Tomko and S. G. Abraham, "Data and program restructuring of irregular applications for cache-coherent multiprocessors," in Proc. ACM Int'l Conf. Supercomputing, Jul. 1994, pp. 214–225.

[18] Intel Intrinsics Guide, https://software.intel.com/sites/landingpage/IntrinsicsGuide/ ed., Intel, 2015, retrieved: February 2016.

[19] M. Corden, Requirements for Vectorizable Loops, Intel, https://software.intel.com/en-us/articles/requirements-for-vectorizable-loops/, 2012, retrieved: February 2016.

[20] Intel® 64 and IA-32 Architectures Optimization Reference Manual, Intel, http://www.intel.com/content/www/us/en/architecture-and-technology/64-ia-32-architectures-optimization-manual.html, Sep. 2014, retrieved: February 2016.

[21] Intel® 64 and IA-32 Architectures Software Developer's Manual. Volume 1: Basic Architecture, Intel, http://www.intel.com/content/dam/www/public/us/en/documents/manuals/64-ia-32-architectures-software-developer-manual-325462.pdf, Sep. 2014, retrieved: February 2016.

[22] G. Goumas, K. Kourtis, N. Anastopoulos, V. Karakasis, and N. Koziris, "Understanding the performance of sparse matrix-vector multiplication," in Proc. 16th Euromicro Intl. Conference on Parallel, Distributed and Network-based Processing (PDP'08), 2008, pp. 283–292.

[23] E. Saule, K. Kaya, and U. V. Catalyrek, "Performance evaluation of sparse matrix multiplication kernels on Intel Xeon Phi," in Proc. Parallel Processing and Applied Mathematics (PPAM 2013), 2013, pp. 559–570.

[24] R. W. Vuduc, "Automatic performance tuning of sparse matrix kernels," Ph.D. dissertation, University of California, Berkeley, 2003.

[25] R. Nishtala, R. W. Vuduc, J. W. Demmel, and K. A. Yelick, "Performance modeling and analysis of cache blocking in sparse matrix vector multiply," University of California at Berkeley, EECS Department, Tech. Rep. UCB/CSD-04-1335, 2004.

[26] F. Wende, "SIMD enabled functions on Intel Xeon Phi CPU and Intel Xeon Phi coprocessor," Konrad-Zuse Zentrum für Informationstechnik Berlin, Tech. Rep. ZIB-Report 15-17, Feb. 2015.

[27] G. Chrysos, Intel® Xeon Phi$^{\mathrm{TM}}$ Coprocessor – The Architecture, https://software.intel.com/en-us/articles/intel-xeon-phi-coprocessor-codename-knights-corner, 2012, retrieved: February 2016.

[28] Intel, Intel® 64 and IA-32 Architectures Optimization Reference Manual, 2014.

[29] AMD64 Architecture Programmers Manual. Advanced Micro Devices, 2013, vol. 3: General-Purpose and System Instructions.

[30] ARM, ARM v8 Architecture Reference Manual, 2010.

[31] M. Flynn, "Some computer organizations and their effectiveness," IEEE Trans. Computers, vol. C-21, 1972, pp. 948–960.

[32] Skylake (microarchitecture), Wikipedia, http://en.wikipedia.org/wiki/Skylake_(microarchitecture), retrieved: February 2016.

[33] GNU GCC, "GCC, the GNU Compiler Collection," retrieved: February 2016. [Online]. Available: https://gcc.gnu.org/

[34] PGI Compilers and Tools, https://www.pgroup.com/, retrieved: February 2016.

[35] T. A. Davis and Y. Hu, "The University of Florida Sparse Matrix Collection," ACM Trans. Math. Softw., vol. 38, no. 1, Nov. 2010, pp. 1:1–1:25.

[36] SPE Comparative Solution Project, Society of Petroleum Engineers, http://www.spe.org/web/csp/.

Use of RBM for Identifying Linkage Structures of Genetic Algorithms

Hisashi Handa

School of Science and Engineering
Kindai University
Kowakae 3-4-1, Higashi-Osaka, Osaka 577–8502, Japan
Email: handa@info.kindai.ac.jp

***Abstract*—Linkage Identification is a quite important in Genetic Algorithm studies. If we found linkage structures, we can improve coding methods, genetic operations. The Restricted Boltzmann Machine (RBM) is adopted to capture linkage structures in this study. The learning data of the RBM consist of data generated by referring to the nonlinearity check of the Linkage Identification by non-Linearity Check. The activation values of the neurons in the hidden layer in the RBM and corresponding learning data are visualized. We can easily find out linkage structures by using the resultant images.**

Keywords–Genetic Algorithms; Linkage Identification; Restricted Boltzmann Machines.

I. Introduction

Linkage Identification plays crucial roles in Evolutionary Computation. The linkage means that a certain tuple of genes appears with a higher rate more than the expected value of a tuple of independent genes [1]. Such genes with higher linkages are supposed to be functionally dependent. Based upon the Building Block Hypothesis [2], several building blocks which have higher linkages are juxtapositionally searched. They are combined with each other by using crossovers implicitly. Therefore, the GA community has been studying this area, since a wrong problem encoding method with strong linkage causes of performance deteriorations [3]. We can elaborate genetic operators if we found the linkage structure in advance. Although Estimation of Distribution Algorithms (EDA) are a systematic approach to solving problems taking into account for problem structures, some EDA, such as the Distribution Estimation Using Markov network (DEUM) still need linkage structures in order to design Markov Networks [4][5].

In this paper, we use RBM to capture the linkage structures. That is, we do not pursue to develop a new mechanism to detect linkage structures. We can detect linkage structure by seeing the visualized image as a consequence of the learning of the RBM.

The organization of the remainder of the paper is as follows: Section II briefly summarizes related works. Section III introduces Linkage Identification by Nonlinearity Check (LINC). We employ the non-linearity check in the LINC to detect the non-linearity in the proposed method. Section IV explains the RBM. Section V describes the proposed method. Preliminary experiments on several benchmark problems are examined in Section VI. Section VII concludes this paper.

II. Related Works

One aspect of linkage identification problems is that they can be regarded as a permutation problem [3]. In genetic algorithms we usually use bit string representation. Therefore, in order to address linkage identification problems, we can solve for an ordering problem of individual representation. The effectiveness of the inversion operator is investigated by Bagley [6], where the inversion operator reverses the order of a partial sub-string in an individual. The idealized reordering operator introduced by Goldberg and Bridges [7] can address the linkage learning problems. Ying-ping showed the performance of the genetic algorithms with the idealized reordering operator can be improved by using tournament selection [8].

Conventional linkage identification studies tackle to devise the mechanism to detect linkages: LINC by Munetomo *et al.* uses non-linearity check among two loci to find out the linkage [9]. His group has been extended to cope with more realistic problems, such as Linkage Identification with Epistasis Measure considering monotonicity (LIEM), and Linkage Identification by non Monotonicity Detection (LIMD) [10][11].

Our main contribution of this paper is to tackle to address of the linkage identification problems by using representational learning algorithms, i.e., the RBM.

III. Linkage Identification by Non-linearity Check

LINC by Munetomo *et al.* focused on the non-linearity among two loci. This paper adopts this nonlinearlity check method in order to generate learning data of the RBM.

Suppose that s indicates an individual whose string length is L. In order to investigate the linkage structures, s is randomly sampled at first. Base upon the individual s, non-linearity is checked as followings: Now, $\Delta f_i(s)$ stands for the fitness difference if the i^{th} locus is changed:

$$\begin{aligned}\Delta f_i(s) &= f(..\bar{s}_i....) - f(..s_i....) &(1)\\ \Delta f_j(s) &= f(.....\bar{s}_j.) - f(.....s_j.) &(2)\\ \Delta f_{ij}(s) &= f(..\bar{s}_i..\bar{s}_j.) - f(..s_i..s_j.), &(3)\end{aligned}$$

where $\Delta f_{ij}(s)$ means the fitness difference if the i^{th} and j^{th} loci are simultaneously changed.

Equation (4) is used if the non-linearity exists among the i^{th} and j^{th} loci:

$$|\Delta f_{ij}(s) - (\Delta f_i(s) + \Delta f_j(s))| > e, \quad (4)$$

where e is a small positive number.

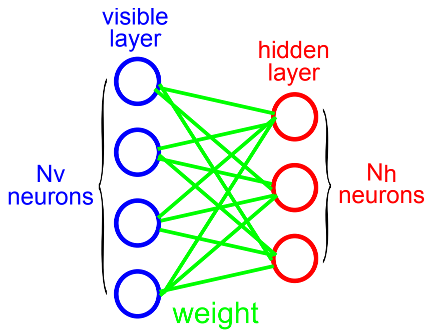


Figure 1. Depiction of Restricted Boltzmann Machine

IV. Restricted Boltzmann Machines

RBM is a two-layered network as shown in Figure 1: The visible layer v and the hidden layer h. Each neuron in the visible layer v is fully connected with all the neuron in the hidden layer h. Note that there are no connections among neurons in the visible layer and among neurons in the hidden layer.

Energy function $E(\mathbf{v}, \mathbf{h})$ can be defined as follows:

$$E(\mathbf{v}, \mathbf{h}) = -\sum_{i}^{N_h}\sum_{j}^{N_v} w_{ij}h_i v_j - \sum_{j}^{N_v} b_j v_j - \sum_{i}^{N_h} c_i h_i, \qquad (5)$$

where N_h and N_v denotes the number of neurons in the hidden layer h and in the visible layer v, respectively. h_i indicates i^{th} neuron in the hidden layer. v_j is j^{th} neuron in the visible layer. w_{ij}, b_j, and c_i are learning parameters for the RBM: weight values, bias values for the visible layer, and bias values for hidden layer. Parameters, such as w_{ij}, b_j, and c_i, are learned by using Constrictive Divergence method.

The joint probability $P(\mathbf{v}, \mathbf{h})$ is represented by using this energy function $E(\mathbf{v}, \mathbf{h})$ in equation (6):

$$P(\mathbf{v}, \mathbf{h}) = \frac{1}{Z}\exp(-E(\mathbf{v}, \mathbf{h})), \qquad (6)$$

where Z is a normalized factor.

For input $\mathbf{v}$, the activation values of the neurons in the hidden layer is calculated as follows:

$$p(h_i = 1|\mathbf{v}) = \sigma(c_i + \sum_{j=1}^{N_v} w_i j v_j), \qquad (7)$$

where σ means Sigmoid function.

V. Proposed Method

A. Overview

This paper uses the RBM to identify the linkages in problems. First, the non-linearity check in (4) is used to detect loci which may have linkages. Such loci information is transformed into learning data. These learning data are learned by using the RBM. After the learning of the RBM, the learned RBM examines the learning data again in order to investigate the activated values for the learning data. The averaged vector of the learning data, such that a certain neuron in the hidden layer is activated, is calculated. An image is generated by using a set of the averaged vectors for all the neurons in the hidden layer. From the image, we can find out if the linkage exists.

```
1:  repeat
2:     An individual $s$ is randomly generated.
3:     for each pair $(i, j) \in A$ do
4:        if rand() $< p$ then
5:           The pair $(i, j)$ is examined the non-linearity check
             in (4).
6:           if non-linearity is found for the pair $(i, j)$ then
7:              A learning data based on the pair $(i, j)$ is ap-
                pended.
8:           end if
9:        end if
10:    end for
11: until The number of learning data reaches to the prede-
    fined number
```

Figure 2. Procedure of generating learning data

```
0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0
0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0
0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
0,0,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,0,0
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,0,0,0,0
```

Figure 3. Learning data generated by the procedure in Figure 2

B. Generation of Learning Data

We assume that in order to solve a given problem to be optimized, i.e., fitness function, a binary encoding method with L bits is used. That is, individuals s are represented by a L bit string. Figure 2 summarizes the procedure of generating learning data, where A denotes a set of all the combinations of two numbers picked from 1 to L without repetition. The “rand()” at line 4 in the procedure is supposed to be random function which returns a random number in $[0, 1]$ over the uniform distribution. A parameter p at line 4 is set to be 0.05 in this study. The number of learning data at line 11 in the procedure is set to be 100,000.

The generation of learning data at line 7 in Figure 2 is described as follows: First, an L bit-string is generated, where all the data in the string is set to be 0. Second, flip the i^{th} and j^{th} data to 1. Figure 3 shows an example of learning data generated by the procedure in Figure 2. Each row denotes each learning data so that this problem seems to be the one of 27-bits length. At the first row, the second and the third loci is set to be 1. Hence, this learning data implies these loci may have a linkage at these loci. This example only shows 10 data while the number of learning data in this study is set to be 100,000.

These learning data are applied to the RBM. The number of neurons in the visible layer N_v is the same as the string length L. The number of neurons in the hidden layer N_h is set to be around 90 % of N_v. These numbers are varied for problem instances; the exact number of N_v and N_h is explained in Section VI.

C. Generation of Averaged Vectors

As depicted in Figure 4, after the learning of the RBM, the learned RBM examines the learning data again in order to investigate the activated values of the neurons in the hidden layer. The input pattern, i.e., each of learning data and the corresponding activated values, is stored.

Suppose that u_i denotes a set of learning data, such that the i^{th} neuron in the hidden layer is activated. In this paper, the hidden neuron is activated if the activated value is greater than 0.5. The average vector $\mathbf{x}_i^*$ is averaged over the entire learning data in the set u_i. Figure 4 shows an example, where the averaged vector is calculated when the second neuron in the hidden layer is activated. The averaged vector $\mathbf{x}_i^*$ is a real-valued vector with the L dimensions. The range of the elements of the averaged vector $\mathbf{x}_i^*$ is [0, 1], since each learning data is a binary vector.

D. Visualization of the Averaged Vector

A binary vector $\mathbf{b}_i = (b_i^1, b_i^2, \ldots, b_i^L)$ is generated from each of the averaged vector $\mathbf{x}_i^* = (x_i^{*,1}, x_i^{*,2}, \ldots, x_i^{*,L})$, where $i = (1, 2, \ldots, N_h)$: Each b_i^j is set to be 1 if $x_i^{*,j} > 0.5$. Otherwise, b_i^j is set to be 0.

These binary vectors $\mathbf{b}_i$ are only used for sorting the averaged vectors $\mathbf{x}_i^*$. That is, these binary vectors are regarded as binary numbers so that these binary numbers are used as a key for the sort. Based upon the key, the $\mathbf{x}_i^*$ is sorted. The sorted $\mathbf{x}_i^*$ is used to generate an image, such that the ping or red color at a certain pixel indicates there may be linkage at corresponding variables.

Figure 5 shows an example of generated image. Each averaged vector is delineated horizontally. Each element of the averaged vector $\mathbf{x}_i^*$ is assigned 10x10 pixels. Hence, the image has the width $L \times 10$, and the height $N_h \times 10$.

As in Figure 5, white color denotes that the corresponding element of the averaged value is 0, which means it has no linkage. Meanwhile, red color means the corresponding element has linkage with the other colored element in the same row. Pink color indicates marginal: strong pink denotes that it tends to have linkage while light pink denotes that it may have linkage.

VI. Experiments

A. Fitness Functions

- **n-trap function** has n-bit linkages. Suppose that The n-trap function is composed of m sub-functions f_t, where $m = L/n$:

$$f_t(\mathbf{s}, k) = \begin{cases} n & \text{if } n_k = n \\ n - 1 - n_k & \text{Otherwise,} \end{cases} \tag{8}$$

where n_k indicates the number of one's in the k^{th} sub-function.
Fitness function f_{trap} can be defined as follows:

$$f_{\text{trap}}(\text{s}) = \sum_{k}^{m} f_t(\text{s}, k) \tag{9}$$

- **6-trap bi-polar function** is similar to the n-trap function where $n = 6$. The sub-function f_b of the 6-trap bi-polar function, however, has two peaks.

$$f_b(\mathbf{s}, k) = \begin{cases} 3 & \text{if abs}(3 - n_k) = 0 \\ 2 - \text{abs}(3 - n_k) & \text{Otherwise,} \end{cases} \tag{10}$$

where n_k is the same meaning as in the n-trap function.
Fitness function f_{6trapBP} can be defined as follows:

$$f_{\text{6trapBP}}(\text{s}) = \sum_{k}^{L/6} f_s(\text{s}, k) \tag{11}$$

- **hierarchical trap function** can be recursively defined: there are a large number of 3-trap-like function, which are connected as in a tree. For explanation, we assume that $L = 27$. We have nine 3-trap-like function f_h^1 at the bottom level as follows:

$$f_h(\mathbf{s}, k) = \begin{cases} 3 & \text{if } n_k = 3 \\ 3 - n_k & \text{Otherwise,} \end{cases} \tag{12}$$

If $n_k = 3$ or $n_k = 0$, such function outputs 1 or 0 to the upper level, respectively. Otherwise, the function output nil.
In the second level, i.e., the upper level of the bottom level. We have three 3-trap-like function. The outputs from the bottom level are used for fitness calculations. Therefore, each 3-trap-like function at the second level is connected to three 3-trap-like functions at the bottom level. The 3-trap-like functions output nil if there is/are (an) output(s) nil from at least one 3-trap-like function at the lower level. Otherwise, 3-trap-like function f_h in (12) is used.
At the top level, there is a 3-trap function. As in the 3-trap-like function at the second level, this function at the top level uses the output from the functions at the lower level, i.e., the second level. This function at the top level also output nil if at least one 3-trap-like function(s) output(s) nil.
The fitness of the hierarchical trap functions is calculated by summing all the outputs of the 3-trap-like function at the top level and the one of 3-trap-like functions at the other level. Note that nil is regarded as 0 for this calculation.

TABLE I. FITNESS FUNCTIONS AND THE NUMBER OF NEURONS IN THE VISIBLE LAYER N_v AND THE HIDDEN LAYER N_h

fitness function	$L(= N_v)$	N_h
5-trap	50	40
six-bipolar	48	40
hierarchical trap	27	24
hierarchical trap	81	72

B. Experimental Settings

As described in Table I, we examined several fitness functions explained the previous subsection. Except for the hierarchical trap function, all the functions are decomposable functions. Table I also summarizes the number of neurons in the visible layer N_v and the hidden layer N_h of the RBM.

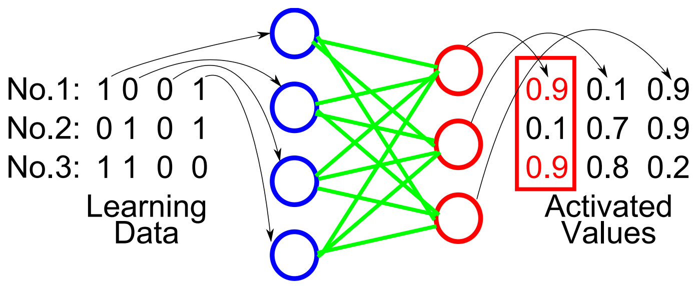


Figure 4. Learning data generated by the procedure in Figure 2

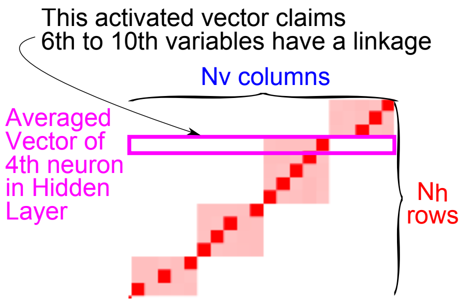

This activated vector claims
6th to 10th variables have a linkage
Nv columns
Averaged Vector of 4th neuron in Hidden Layer
Nh rows

Figure 5. Example of visualization

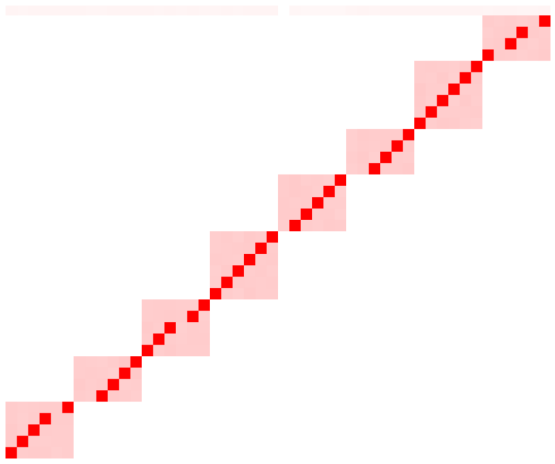


Figure 7. Experimental results for 6-trap bipolar function

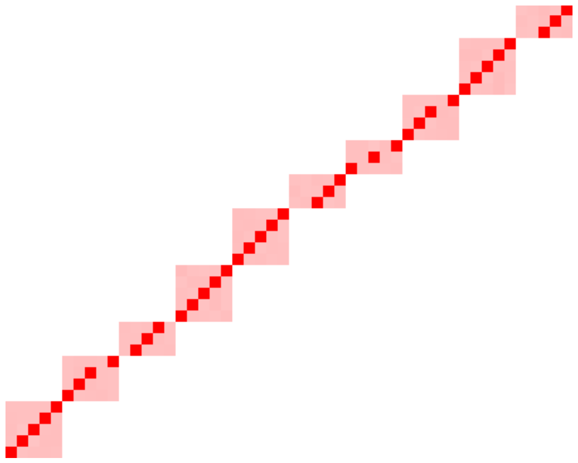


Figure 6. Experimental Results for 5-trap function

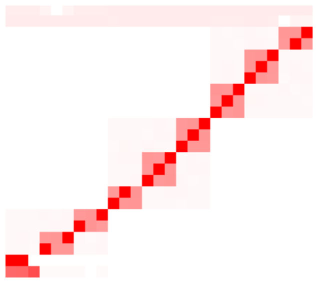


Figure 8. Experimental results of 27-bit Hierarchical Trap Function

C. Experimental Results of Decomposable Functions

Figures 6 and 7 show the experimental results for the 5-trap function and the 6-trap bipolar function, respectively. There are ten and eight linkage clusters in these functions. Even though the non-linearity check examined in this paper concerns with only two bits, the proposed method can capture such linkage structures well.

The first averaged vector in Figure 7 depicted lighter pink for all the variables. Such lighter pink patterns are often appeared.

D. Experimental Results of Hierarchical Trap Function

Figures 8 and 9 show the experimental results of the non-decomposable function, i.e., the Hierarchical Trap function. We examined 27-bit Hierarchical Trap Function — three level problem —, which is explained in subsection VI-A. Moreover, 81-bit Hierarchical Trap Function — four level problem — is also examined. These figures show that the proposed method can find out linkage structures for the bottom level and the second level. For the second level, we can see that the lighter pink areas are for each tuple of corresponding nine variables. However, the third level and the fourth level (for 81-bit problem) cannot extract linkage structure at all. The reason for this is that it is unable to capture such dependence structure by only two bits which are examined in the non-linearity check in this paper. Therefore, we think that non-linearity check with the higher order is needed.

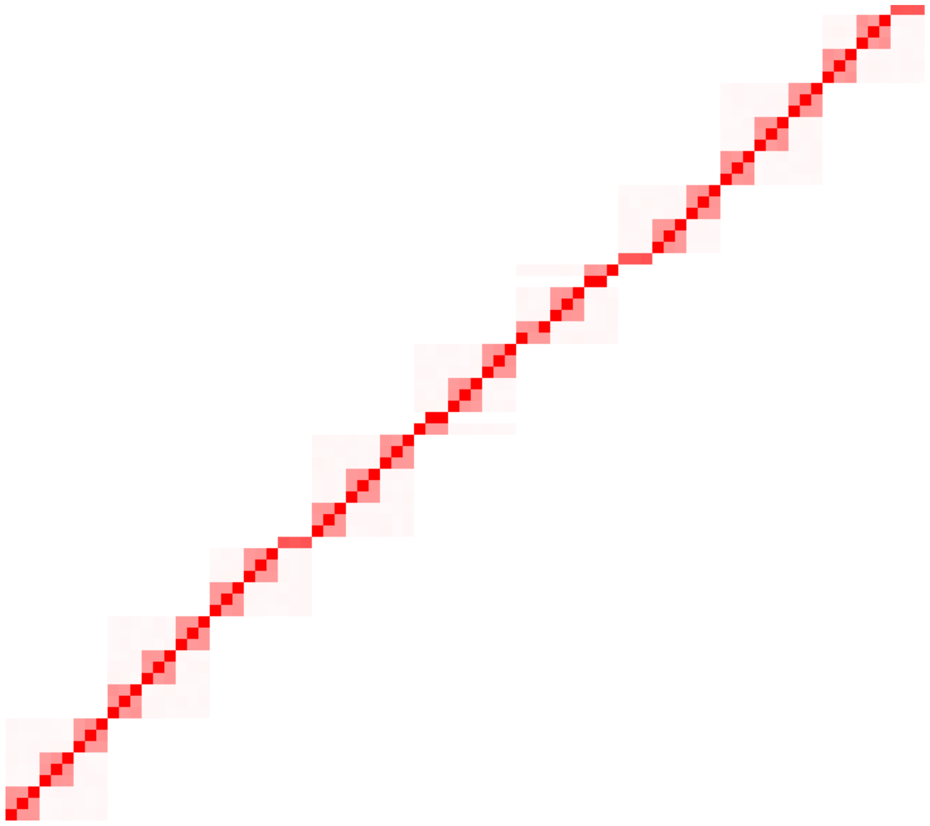


Figure 9. Experimental results of 81-bit Hierarchical Trap Function

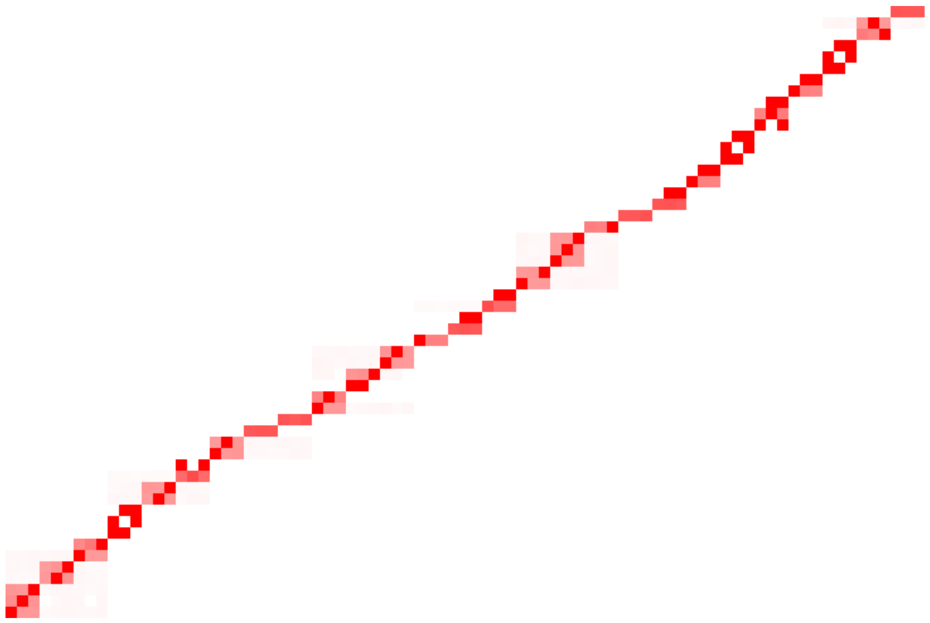


Figure 10. Experimental results of 81-bit Hierarchical Trap function by the Deep Boltzmann Machine

E. Comparison with "Deep" Structures

Recently, the RBM is used in Deep Learning, which is a part of Deep Belief Net., or Deep Boltzmann Machine. We examine the effect of deep structure in neural networks for linkage detection. In Figure 10, the experimental results of the DBM are shown. We employed 81-bit Hierarchical Trap function for this comparison. The network structure of the DBM, i.e., the number of neurons in each layer, is 81, 72, 63, and 54. For comparison, we constitute the RBM such that the number of neurons in the visible layer and the hidden layer are 81 and 54, respectively. The experimental results

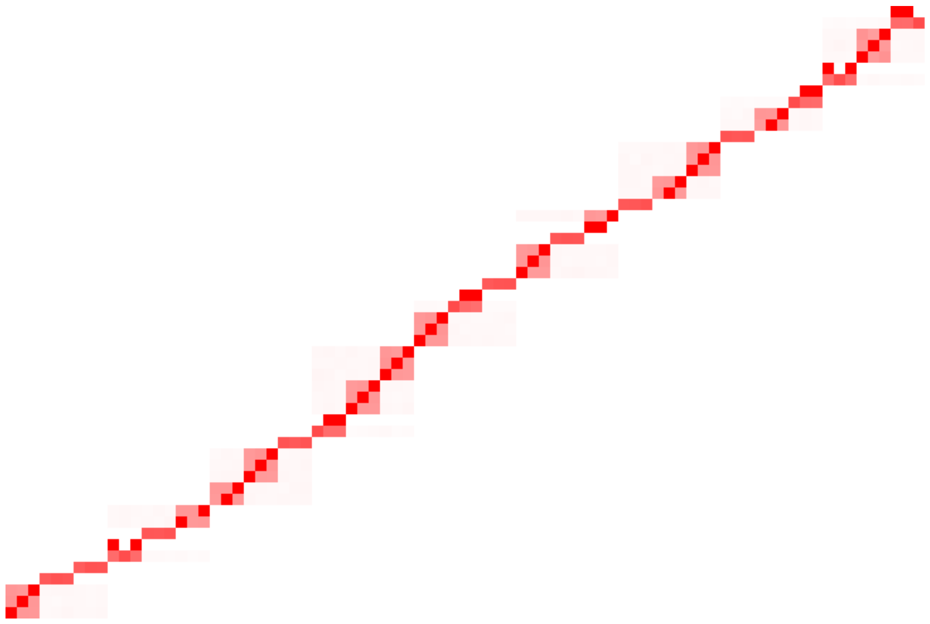


Figure 11. Experimental results of 81-bit Hierarchical Trap function by a single RBM with 54 neurons in the hidden layer

of the RBM are shown in Figure 11. It seems that there are no significant differences between these figures. Even if deep structures are employed, we cannot detect linkage structures in the third and fourth levels. We might need to examine more various network structures for the DBM. However, we think that the input for the RBM and the DBM are modified in order to capture linkage structure, i.e., non-linearity check so that such higher dependency are disregarded at the learning data generation.

VII. Conclusion

In this study, we employed the RBM to detect linkage structures in fitness function which are used in Genetic Algorithms. In order to constitute the learning data for the RBM, non-linearity check used in LINC is used. We also introduced a visualization method by using sorted averaged vectors. We can easily find out by using the resultant images if the linkage structures exist.

Future works are summarized as follows: We extend non-linearity check mechanism to cope with a higher order of linkage detection. We think that more bits should be simultaneously checked. On the other hand, we would like to use the deep learning in this study. For this purpose, the representation of learning data must be considered.

Acknowledgment

This work was partially supported by the Grant-in-Aid for Scientific Research (C) and the Grant-in-Aid for Young Scientists (B) of MEXT, Japan (26330291, 23700267)

References

[1] D. R. Newman, "The Use of Linkage Learning in Genetic Algorithms," http://www.ecs.soton.ac.uk/ drn05r/ug/irp/, Last Update: September 2006.

[2] D. E. Goldberg, "Genetic Algorithms in Search, Optimization, and Machine Learning," Addison-Wesley, 1989.

[3] C. Ying-ping, Y. Tian-li, S. Kumara, and D. E. Goleberg, "A Survey of Linkage Learning Techniques in Genetic and Evolutionary Algorithms," Technical Report IlliGAL Report, 2007.

[4] P. Larranaga and J. A. Lozano (Eds.), "Estimation of distribution algorithms: A new tool for evolutionary computation," Kluwer Academic Publishers, 2002.

[5] S. K. Shakya, J. A. W. McCall, and D. F. Brown, "Updating the probability vector using MRF technique for a Univariate EDA," Proceedings of the Second Starting AI Researchers' Symposium, volume 109 of Frontiers in artificial Intelligence and Applications, 2004, pp. 15-25.

[6] J. D. Bagley, "The behavior of adaptive systems which employ genetic and correlation algorithms," Ph.D. dissertation, University of Michigan, Ann Arbor, MI, 1967.

[7] D. E. Goldberg and C. L. Bridges, "An analysis of a reordering operator on a GA-hard problem," Biological Cybernetics, vol. 62, 1990, pp. 397-405.

[8] C. Ying-ping and D. E. Goleberg, "An analysis of a reordering operator with tournament selection on a GA-hard problem," Proceedings of Genetic and Evolutionary Computation Conference 2003 (GECCO-2003), 2003, pp. 825-836.

[9] M. Munetomo and D. E. Goldberg, "Designing a genetic algorithm using the linkage identification by nonlinearity check." Technical Report IlliGAL Report No.98014, University of Illinois at Urbana-Champaign, 1998.

[10] M. Munetomo and D. E. Goldberg, "Linkage identification by non-monotonicity detection for overlapping functions," Evolutionary Computation, Vol. 7, No. 4, 1999, pp. 377–398.

[11] M. Munetomo, "Linkage identification with epistasis measure considering monotonicity conditions," Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution and Learning, 2002, pp. 550-554.

[12] A. Fischer and C. Igel, "An Introduction to Restricted Boltzmann Machines," Proc. CIARP 2012, LNCS 7441, 2012, pp. 14-36.

[13] R. Salakhutdinov and G. E. Hinton, "Deep Boltzmann machines," *Proc. JMLR Workshop and Conference Proceedings: AISTATS 2009*, vol. 5, 2009, pp. 448-455.

Post Deployment Secure Key Management in Wireless Ad hoc Networks

Paul Loree and Kendall Nygard
Dept. of Computer Science
North Dakota State University
Fargo, ND USA
Email:{paul.loree, kendall.nygard}@ndsu.edu

Abstract— **Providing secure communication between nodes in mobile ad hoc networks is critical in many applications. In this paper, we present our work on a key distribution scheme for mobile ad hoc networks. Unlike traditional key distributions which establish encryption keys prior to node deployment we devise a post-deployment key distribution scheme to allow nodes a higher chance of connectivity with neighboring nodes. At the same time, nodes are able to be highly mobile within the network while reducing the need for excessive storage of encryption keys by each node in the network.**

Keywords-mobile ad hoc networks; wireless ad hoc networks; network security; key management; key distribution; secure communication; WANET; MANET; VANET.

I. INTRODUCTION

Wireless technology has advanced rapidly in recent years. New advances in low powered microcontrollers, such as Atmel [1] and ARM [2], and miniature computers [3] [4] have allowed for wireless enabled devices to be networked together to provide novel and interesting devices both commercially and by hobbyists. One of the promising networking techniques advanced in recent years are wireless ad hoc networks (WANETs). WANETs are organized as a decentralized network which operate over a wireless medium to provide communication services among different devices. Unlike an infrastructure network, WANETs are self-organizing, whereby the nodes in the network cooperate to provide the routing of data. One of the benefits of using wireless WANETs is robustness to node failure. Nodes participating in an ad hoc network join and leave the network randomly while not disrupting the network and its ability to route messages towards the destination. WANETs have several applications, such as in personal area networks (PANs), sensor networks (WSNs) vehicular WANETs (VANETs), and mobile WANETs (MANETs).

WANETs may also consist of nodes with different capabilities. For example, an ad hoc network may consist of several battery operated low powered devices, such as sensor nodes, and more powerful nodes, such as a laptop or tablet computer, which aggregate and process the sensor node data. MANETs and other WANETs have applications in many areas, including military surveillance, emergency incident response, industrial and agricultural monitoring, and in-home automation. For example, a military deployment may utilize a UAV-MBN network which consists of three levels of ad hoc networks operating together. At the lowest level the military may distribute sensor nodes into a battlefield environment which communicate information back to devices being used by troops on the ground through an ad hoc network [5]. The devices used by the troops are then connected through a localized mobile backbone network consisting of ground vehicles in the middle layer. At the highest level are unmanned aircraft which are used as a backbone to connect the ground vehicles over rough terrain and distances. Other uses of an ad hoc network include extending the range of existing infrastructure networks to allow additional devices to communicate via the Internet while out of range of infrastructure equipment, or connecting devices to the “Internet of Things”. Securing the communication between the nodes in these networks is essential for high resistance to attackers who want to disrupt communication or gather information from the network.

The heterogeneous nature of these networks presents unique challenges for providing secure communication among all parties in the network. Resources available to devices found in the network may vary between devices and have limited capacity. Sensor network nodes, for example, may be limited by computational power, transmission range, and rely on battery resources for a power source. However, a laptop with an external wireless card connected to a power source does not have these limitations. This creates a serious problem for heterogeneous WANETs, as some devices may allow existing security techniques to be used, but they may not apply to other devices in the network.

To provide confidential communication in a network, key distribution must take place. A simple solution would be to preload a single key for encryption on each node. However, this approach has a drawback, because an attacker would only need to compromise one node in the network. Since WANETs rely on the cooperation and trust of all nodes in the network to handle routing, the compromised node is likely to receive a significant amount of data that may contain confidential information. Another simple solution using current technologies is to use a transient key based on a passphrase, as seen in 802.11 infrastructure networks. While this allows for easy implementation in the 802.11 protocol for securing the traffic on the network, not all WANETs may be able to utilize these security mechanisms. Additionally, nodes joining the network must know the predefined passphrase to connect to the network. This may not be possible in all situations, since the passphrase may not be known. For example, a VANET may consist of vehicles made by different manufacturers using their own passphrases for vehicles to form VANETs. This would reduce the usage of the VANETs being formed. To be effective in the unique environment of MANETs, some method of encryption key distribution must be devised that does not rely on existing methods.

This paper is organized as follows: in Section II, we present related works in key management schemes for MANETs. Section III discusses our key distribution scheme. Section IV discusses our preliminary analysis and experimentation results of our proposed key distribution scheme. We conclude the paper in Section V and discuss our future plans for this research.

II. Related Works

Many key distribution schemes have been proposed in recent years for providing encryption keys to nodes in various types of WANETs. In this section, we discuss several of these proposed methods.

One of the first schemes proposed for establishing secure communication in WANETs was described in [6]. Eschenauer and Gligor's scheme was designed to be deployed in WSNs. Each node randomly selects a subset of keys from a key pool prior to being deployed in the network. After deployment, the nodes establish secure communication if two nodes share a common key. However, this scheme may not guarantee secure communication can be established since the scheme is based on the probability that nodes share a common key. To increase the probability a larger subset of keys must be preloaded which also increases vulnerabilities if a node is captured as more keys are exposed to the attacker. Chan et al. improved upon the security of this scheme in [7] by proposing that each node must have at least *q* keys in common between communicating nodes at a cost of additional memory.

In [8] and [9], two independent research studies proposed predistribution schemes which generate a set of key spaces and preload each node with a subset of the key spaces, known as keying shares. The difference in their approaches exist in the underlying mathematical framework used. In [8] the scheme uses a key space approach found in [10], while in [9] the authors generate key spaces based on the work found in [11]. Both allow communication to be secured between nodes if they share the same key space, but they rely on the probability of two nodes sharing keying information after deployment. This research expands on the use of keying shares as described in [10] and [11] to provide the generation of encryption keys after nodes have been deployed.

In [12], Hai-tao divides the network into zones based on the connectivity of nodes in a general area and a CH is selected to manage keys for each zone. The cluster key is created at the CH by nodes signing a nonce with their preloaded private key. Each of the nodes is then provided a shadow of the key and reconstructs the cluster key from the signatures of its neighbors.

PushpaLakshmi et al. proposed an agent based composite key management scheme in [13]. Nodes are partitioned into clusters in the network and a CH is selected based on its trustworthiness and probability the node will remain in the network. In their scheme, a fuzzy logic algorithm is used to determine the trustworthiness based on the node's successful routing statistics. Each node is also assigned a public certificate by an offline CA prior to joining the network and CHs are initially selected by an administrator. Additionally, a subset of nodes in the deployment will be used for creating partial private keys for new nodes to join the network. When a node joins the network the node generates its public key and registers with the CH that assigns a unique ID to the node. The CH then assembles the private key for the new node. The main drawbacks to this approach are that an offline CA is required, the initial CH nodes must be assigned by an administrator, and CHs know the private keys for joining nodes.

Liu et al. proposed an in situ key management scheme in [14] which establishes keys in homogeneous network after deployment. Their scheme elects nodes as CHs in a wireless sensor network to act as key distribution centers in the network based on a predetermined probability factor. When a node is elected, it generates a key space and distributes keying shares to nodes nearby which request keying information. However, their scheme was designed to be used in a homogeneous network and allowed the CHs to die after distributing the keying information.

In [15], Loree et al. proposed a similar scheme that extended on the work proposed in [14] by introducing a scheme designed to more efficiently provide keying information to a heterogeneous sensor network which used more powerful nodes to act as CHs. This work was further investigated in [16] and is the basis for this work in MANETs.

Zhao et al., [17] also proposed a scheme that uses a CA to provide public key encryption keys to wireless mesh networks. Unlike the previous scheme however, the CA is part of the infrastructure backbone of the network and nodes register when joining the network. Additionally, the CA must remain in the network which may not be suitable in a MANET environment.

Boukerche et al. proposed a key management scheme for MANETs in [18]. Their scheme provides both asymmetric and symmetric encryption to nodes in the network. Their work attempts to address the issue that a CA may not be present in the network to after deployment. They do require a CA to exist to distribute public/private keys to nodes prior to the network being formed. Additionally, each node is required to store both public keys and session keys of its neighbors. Lastly, since each message establishes a session between nodes, significant communication overhead occurs due to session key being established between nodes in the route.

In [19], Chauhan and Tapaswi describe a key management scheme that provides both asymmetric and symmetric encryption. In their scheme several nodes are preloaded with a public/private keys and a function to create key pairs for other nodes in the network. Each node stores either a symmetric key or a public key for use with each of the nodes in their cluster. Nodes are allowed move between clusters but must be validated by the previous CH before the node is given new keys. However, this is a drawback in this scheme when a node joins the network because the CH must know about the node prior to being added. Another drawback to this scheme is that nodes may only join clusters they are assigned to prior to setup or each CH would require information about all of the nodes in the network.

Lu et al., proposed a certificateless key distribution scheme in [20]. Their scheme distributes parts of a master secret key to several key generating centers (KGC) that provide keying information to the nodes. In their approach a several KGCs must be contacted by a node to receive keying information, reducing the chance a single KGC is compromised the network if attacked. In their approach each node is able to calculate its neighbor's public key for secure communication with the neighbor's ID and the current key phase's salt value.

In [21], Dahshan and Irvine propose a MANET key management scheme which distributes threshold cryptography keys. In their scheme the nodes in the network consist of two types of devices, CA trusted nodes authenticated prior to deployment and non-trusted nodes which joined after deployment. Nodes establish a connection-orientated route to the destination only through CA trusted nodes. Both source and destination nodes may then either accept or deny the route to be used. Nodes are also able to individually revoke certificates if they believe it has been compromised. However, each node must keep a public certificate for all nodes in a route which may not scale well especially if nodes are required to persistently store all received certificates. If nodes are allowed to delete the certificates there will be significant communication overhead when establishing a route for each message and nodes would no longer be able to revoke keys. Lastly, only CA authenticated nodes are used in a route and the network may not be fully connected since nodes are mobile and may have moved out of range of CA authenticated nodes. In order to ensure that nodes stay connected in their scheme enough statically located CA authenticated nodes must be placed knowingly in the topology of the network prior to deployment.

In [22], Seghal et al. presented analysis of security issues in MANETs. Their work provided analysis of the existing problems facing three key areas of MANETs, key management, ad hoc routing, and intrusion detection. They state two main problems are faced when dealing with key management in MANETs. The first problem is it is difficult for nodes in the network to determine if nodes in the network have revoked a certificate used by a node in the network. A second problem is that nodes may be in different trust hierarchies and their certificates may not be valid across different levels in the network. They propose the solution to this is to use a trusted third party (3P) or global password authentication for all nodes to use to gain access. Both of these however pose drawbacks since MANETs may be created in places with limited infrastructure or a 3P CA would be infeasible. Global passwords also provide limited security since clients with malicious intent can gain access to this information easily. This demonstrates the need for a distributed key distribution system that can be efficient to deploy to nodes joining the network without compromising the security of existing or future clients in the network.

III. WANET Key Distribution Scheme

In this section, we describe our key management scheme for providing keys for symmetric encryption to nodes in a WANET. We expand on our work in [15] and [16] for support of WANETs. In our scheme nodes are partitioned into clusters and CHs are elected post-deployment to provide keying information for nodes to generate encryption keys. A secret key is generated between two nodes using key space models discussed in [11] and [10]. These key spaces are generated using either a bivariate symmetric n-degree polynomial [11] or symmetric public and private matrices of $(n+1)\times(n+1)$ dimensions [10]. The coefficients of the polynomial or the elements of the matrices are generated after the network is deployed. Keying information is distributed to the nodes in a cluster by a CH. By using the keying information, each node is able to create secret keys for symmetric encryption between themselves and their neighbors using their IDs as input.

A. Key Space Models

Our scheme will operate under both key space models described in [11] and [10]. In [11], the authors utilize a symmetric n-degree polynomial. In the first key space model we use the bivariate symmetric n-degree polynomial such that

$$f(x,y) = f(y,x) = \sum_{i,j=0}^{n} a_{i,j} x^j y^j \qquad (1)$$

over a finite field F_r, where r is a large prime number that can be used for cryptographic keys. A key for symmetric encryption can be found by two nodes in the network that share the same coefficients, a_i, the key space, by exchanging IDs and computing (1). In our scheme, the CH generates a set of functions, F, and each node in the cluster is provided with one of the generated functions as its keying information. When creating a secure link between two nodes in the cluster the nodes send their calculated coefficients for their function using their ID, x. This allows the receiving node to compute the same key by inputting their ID, y to create the secret key.

In [10], the authors proposed a similar method that uses matrices instead of a polynomial. Nodes exchange column values of a public matrix along with their ID to compute a key. One benefit to using this key space method is that communication overhead can be reduced by using a generation matrix, e.g. Vondermonde matrix [8], where a seed value is used to compute a column. This however increases the computational overhead by $n-1$ modular multiplications.

A valuable property in both of these models is that they are both n-collusion resistant. That is, where less than n nodes using the key space remain uncompromised, the key space itself remains secure and new keys generated by the key space cannot be determined using the information contained in the compromised nodes as shown in [11] and [10]. Furthermore, both of these models have low storage overhead as shown in [14].

B. Assumptions

In our work, we assume no prior knowledge of node placement before deployment. We also assume that nodes are mobile and may move between clusters in the network. Nodes may belong to several clusters if they are within transmission range of a CH. CHs are also assumed to operate normally in the network in addition to CH responsibilities. We also assume each node is loosely time synchronized within the network. Furthermore, the network supports an ad hoc routing protocol such as Ad hoc On-demand Distance Vector (AODV) or Dynamic Source Routing (DSR). Each node is preloaded with various parameters prior to joining the network described in Table 1.

TABLE I. PRELOADED PARAMETERS

λ	Maximum number of nodes in a cluster
n	Degree of polynomial or matrix dimensions for generating key spaces
ID	Unique ID
r	Large prime for generating key space over finite field
t_{wait}	Maximum bootstrapping wait time
$t_{threshold}$	Minimum time period before rekeying for new node joins
t_{limit}	Time period for each key share in network
TTL	Time to Live
p, q	Large primes needed for Rabin's cryptosystem
B	Predefined padding for Rabin's cryptosystem

Each cluster has a maximum of λ nodes. Since the key space models are n-collusion resistant the optimal value for the cluster size should be less than or equal to the value of n. Each node contains a unique ID and large prime number, r, to be used to generate key spaces over a finite field, F_r. Each node is also preloaded with three threshold values. These thresholds are used to define the maximum wait time, t_{wait}, before a CH announces it has generated keying information, the minimum amount of time a node is allowed to join before rekeying, $t_{threshold}$, and a key expiration time limit, t_{limit}. Additionally, the maximum time to live for packets being broadcast from the CH is defined by TTL. The last three values are optional if the network is assumed to be insecure at startup then these parameters are used for Rabin's cryptosystem as described in the next section.

C. Rabin's Cryptosystem

In our scheme we assume that the network may be insecure from eavesdropping. In this network model we use Rabin's Cryptosystem [23] to temporarily secure the transmissions between the CH and joining nodes while distributing keying information. This allows new nodes to join without requiring a CA to preload public/private key pairs to new nodes joining the network.

Rabin's cryptosystem is a computationally asymmetric encryption algorithm that uses two large prime numbers to create a public key $n = pq$. The CH sends n to cluster nodes to encrypt a temporary session key, k. The nodes in the network randomly generate k to encrypt the exchange of keying information. Using Rabin's algorithm this session key is encrypted as $E_n(k||B) = (k||B)^2 \, mod \, n$, where B is a preloaded bit pattern. B is required to allow the CH to decrypt the message successfully using Rabin's algorithm. In addition to the session key the node also sends its ID to the CH. We use this cryptosystem because it is computationally cheap for encryption but as computationally expensive as RSA to factor the two large prime numbers, p and q from n.

D. Cluster Head and Key Distribution Algorithms

In this subsection, we describe two algorithms used by CHs to generate key spaces and distribute keying information to nodes in the cluster. The first algorithm we discuss is the Cluster Head Algorithm illustrated below in Figure 1.

In our scheme we elect CHs using a predefined probability threshold value. An alternative would be to elect nodes based on their performance capabilities in the network. If performance capabilities are measured, the CHs can be selected based on several metrics such as computational performance, battery life, wireless medium interfaces on the device, number of neighboring nodes within a specific hop distance, or transmission range.

Once elected, the CH generates a key space based on one of the key space models previously described on Line 2. On line 3, the CH generates a random wait time, w less than the preloaded maximum wait time, t_{wait}. The CH then listens for other nodes that may have also elected themselves to be the CH for an area in the network as shown in the loop starting on line 4. This is to reduce collisions between two nodes in an area of the network that may be simultaneously broadcasting an announcement that they have keying information available and to let the first node announcing it has keying information available to become the CH. Once the announcement has been broadcasted the CH starts the distribution process as illustrated below in Figure 2.

Figure 2 below describes our Key Distribution Algorithm. On line 2, the CH announces its ID to the network to notify nodes within TTL hops that keying information is available. After sending the announcement the CH will start a loop on Line 3 and accepts requests from other nodes in the network within range of its broadcast for keying information. Nodes send requests to the CH using unicast messages. This loop will continue to provide keying information as long as the cluster size limit, $\lambda - 1$, has not been reached. The CH will also save one keying information share for itself so it remains in the cluster it has formed. When the CH receives a request for keying information the CH selects an unused keying information share as shown on Line 6. The CH will also mark

```
Cluster Head Algorithm
 1: function runClusterHead ($\lambda, t_w$)
 2:     $keys$ ← genKeySpace()  // construct key space for $\lambda$ nodes,
                                   store one for self
 3:     $w$ ← random($0 < t_w$) // random wait time
 4:     while $w > 0$ do
 5:         listen for broadcasts from neighboring nodes that
            have elected themselves as cluster heads
 6:         if BROADCAST heard
 7:             RequestKeyInfo() // get keying info
 8:             exit
 9:         end if
10:         elapse($w$)
11:     end while
12:     KeyDistro() // Fig. 2
13: end function
```

Figure 1. Generates Keying Information

```
Key Distribution Algorithm
1: function KeyDistro($\lambda$, ID, keys)
2:     BROADCAST(ID) // broadcast to all nodes within TTL hops
3:     keyed := 1
4:     while keyed < $\lambda - 1$ do
5:         if received(RequestKeyInfo(ID_k))
6:             KeyingInfo ← an unused key share
7:             usedKeys ← usedKeys ∪ {KeyingInfo}
8:             send(ID, $k$(KeyingInfo)) // $k$ if using Rabin's
9:             keyed := keyed + 1
10:        end if
11:    end while
12: end function
```

Figure 2. Distributes Keying Information

the used keying information so that it will not provide this information to another node in the cluster as shown on Line 7. On Line 8, the CH will send an unused keying information to the requesting node using a unicast message. The distribution algorithm runs until all keying information has been used. As long as keying information remains available the cluster may accept new nodes joining the network.

E. Cluster Joining Algorithm

In this subsection, we describe our algorithm for forming clusters and nodes joining a cluster in the network after the network has been established.

Figure 3 shows our Cluster Joining Algorithm. When a node joins a cluster it requests the remaining time until the rekeying phase, as shown on Line 3. Only the CH will respond to this request with a broadcast, so existing nodes are informed of the new node and the remaining time before rekeying the cluster begins. If there is no response, then a cluster does not exist and the node will begin the election process to become a CH, as previously described in Figure 1 and the function exits. If there is a response from a CH and the $t_{remaining} \leq t_{threshold}$ the re-election and rekeying event will be started early and the node will join the newly formed cluster during this phase. During the re-election process nodes will generate a probability or benchmark score on Line 9 and announce their score to their area in the network. The node with the highest value will then become the new CH for the cluster starting on Line 12. Once a node has been elected as a CH it will start generating key spaces as shown on Line 20. Nodes not elected will then request keying information once a CH broadcasts an announcement as described in the previous subsection. Nodes within range of more than one CH will also request keying information from each of the CHs it receives a broadcast from. This allows nodes which are lying on the fridges of a cluster to join multiple clusters and provide routing paths between clusters.

If the value of $t_{remaining} > t_{threshold}$ then the node joining the cluster will request the keying information from the current CH as shown on Line 25. Security can be improved by setting $t_{threshold} = t_{limit}$ since a new node will force rekeying. However, if the network is expected to have a lot of nodes moving into and out of the cluster than this may

```
Join Cluster Algorithm
1: function joinCluster ($t_{threshold}$)
2:     $t_{remaining}$ := 0
3:     $t_{remaining}$ := RequestKeyTime ( ) // get remaining time
                          of current key time from
                          current cluster head
4:     if($t_{remaining}$ == 0)
5:         runClusterHead(λ, $t_w$) //initialization of network
6:         exit
7:     end if
8:     if($t_{remaining}$ <= $t_{threshold}$)
9:         $score_{new}$ ←calcScore( ) // score node capabilities
                          (probability, cpu, storage, batt,
                          RSSI, node connectivity)
10:        $chScore${} ← $scoreRequest$( ) // Broadcast
                          request for cluster node scores
11:        BROADCAST score
12:        foreach $score$ in $chScore$
13:            if($score_{new} > score$)
14:                $newClusterHead$ ← true // elect self
                          as new cluster head
15:            else
16:                $newClusterHead$ ← false // remove
                          election if another node is better
17:            end if
18:        end foreach
19:        if($newClusterHead$)
20:            runClusterHead( ) // become cluster head and
                          generate keying information (Fig. 1)
21:        else
22:            RequestKeyInfo( ) // request keying
                          information from new cluster head
23:        end if
24:    else
25:        RequestKeyInfo( ) // request keying information
                          from existing cluster head
26:    end if
27: end function
```

Figure 3. Joins or forms network clusters

cause unwanted computational and communication overhead.

Additionally, the CHs will restart the keying process once the t_{limit} threshold value has been met. At this time either the node may generate new keys for their cluster or restart the election process to see if a new CH is available. To reduce the load on the previous CH the current CH may also be set to not elect itself again for a period of time and a round robin approach may be used to elect new CHs.

IV. KEY DISTRIBUTION ANALYSIS

In this section, we provide a performance analysis of our scheme. First we re-examine the security performance of the scheme. As shown in [10] and [11], both key space models require at least n nodes in a cluster to be compromised before all of the generated keys for symmetric encryption used in the key space can be calculated. We assert that to increase the security of these key space models that the degree should be at least $n \geq \lambda$. If the number of nodes in each cluster is the same as the degree of the polynomial or the size of the matrices, this will ensure that only if all nodes in a cluster are compromised would the key space be compromised. At this point the attacker would already have access to all keys within the cluster. If $n > \lambda$, then determining any other keys generated in the future from the key space would be impossible, which keeps the future cluster nodes secure. Additionally, this will allow new nodes to join existing clusters before the rekeying process to starts or until all key shares have been distributed.

Next we consider storage requirements for nodes in the network for storing keys for symmetric encryption. Each node will be required to store a key for itself and any node it has received IDs from within its transmission range. If a node has joined k clusters then these nodes would be required to store at most $k\lambda$ keys for symmetric schemes. However, we expect most nodes in the network to only store the nodes within their immediate transmission range and only be connected to one cluster at a time except for nodes along the edges of clusters which join adjacent clusters to allow for communication between clusters. Additionally, nodes will store keying information for the key space. In [14], it is shown that storage requirements for keying information are close to $(n+1)\log r$ in the polynomial based model and $(n+2)\log r$ in the matrix model if the matrix selected for G allows the columns to be seeded.

Finally, we discuss the communication overhead of our scheme. In the following equations let N be the number of nodes in the network, C be the number of elected CHs and λ the number of nodes in each cluster. Below in (2) we show the total number of CH announcement broadcast messages transmitted by CHs in the network during the distribution phases of the scheme. In a network consisting of C clusters each elected CH, E_i, sends a broadcast message to their neighboring nodes. This announcement broadcast message is then retransmitted by at most λ nodes in each cluster while the $TTL > 0$.

$$Broadcasts = \lambda \cdot \sum_{i=1}^{C} E_i \qquad (2)$$

Once the announcements from the CHs have been transmitted to nodes within TTL range of elected CHs, each node within range joins the CH cluster. The total number of messages needed to distribute this information is be shown below in the (3). In order to join the cluster each node which receives the announcement replies by sending a unicast message requesting a keying information from the key space generated by the CH. This requires each node, R_j, up to λ nodes in a cluster to send a unicast message back to the CH. Each unicast message sent is routed up to TTL times in order to reach the CH. This process occurs in each of the C clusters in the network. Each CH then replies to up to λ nodes with a unicast message to each of the requests for keying information nodes.

$$Distributions = 2C \cdot \sum_{j=1}^{\lambda} (R_j + TTL) \qquad (3)$$

During the election and rekeying process each node in each cluster broadcasts its probability or benchmark score in order for the new CH to be selected. The total number of messages this requires is shown in (4) below. Each node in the network, S_k, broadcasts its score value which is then retransmitted a maximum TTL times by its neighbors.

$$Scores = \sum_{k=1}^{N} (S_k + TTL) \qquad (4)$$

The average number of messages sent per each node in our scheme can be calculated by (5) below which is the sum of the previous totals divided by the number of nodes in the network, N.

$$Average = \frac{Broadcasts + Scores + Distributions}{N} \qquad (5)$$

Limiting the average number of messages transmitted over the network by each node is important to increase the life expectancy of nodes relying on battery power.

V. Conclusion

We devised and analyzed an efficient secure key distribution scheme for WANETs. Our proposed scheme has strong security against attackers attempting to discover the shared secret keys used for symmetric encryption by being n-collusion resistant, which requires attackers to compromise an entire cluster of nodes before keying information can be discovered. We also provide efficiency in our scheme in terms of computational, storage and communication overhead in order to increase the life expectancy of battery powered nodes placed in the network.

Unlike existing schemes, our method does not rely on prior knowledge of the network topology and can establish keys for symmetric schemes to be used between nodes after deployment. Additionally, our scheme does not require centrally administered CAs to be used in the network to distribute keys to nodes prior to the nodes joining the network. Furthermore, our scheme allows all nodes to be mobile in the network and does not require nodes to remain statically assigned to locations in the network for handling routing and key distribution, which may be difficult in many situations, such as in MANETs.

In the future, we plan to empirically evaluate our scheme by using simulations to compare our scheme with competing existing schemes for connectivity between nodes. Additionally, we plan to investigate adding authenticity and integrity mechanisms into the scheme to improve the security of the network not covered under confidentiality provided by encrypted communication. We also plan to simulate our scheme against these other methods while under various attacks unique to WANETs such as Sybil, black hole, and wormhole attacks. Finally, we plan to compare our scheme against other methods for routing messages across the network between mobile nodes while under normal network conditions and while under attack.

References

[1] "Atmel Corporation," [Online]. Available: http://www.atmel.com/. [Accessed November 2015].

[2] "ARM," [Online]. Available: http://arm.com/. [Accessed November 2015].

[3] "Arduino," [Online]. Available: http://www.arduino.cc/. [Accessed November 2015].

[4] "Raspberry Pi," [Online]. Available: http://www.raspberrypi.org/. [Accessed November 2015].

[5] H. Deng, R. Xu, J. Li, F. Zhang, R. Levy, and W. Lee, "Agent-based Cooperative Anomaly Detection for Wireless Ad Hoc Networks," in 12th International Conference on Parallel and Distributed Systems, 2006, pp. 1-8.

[6] L. Eschenauer and V. D. Gligor, "A Key-Management Scheme for Distributed Sensor Networks," in 9th ACM Conf. Computer and Communications Security, Washington, DC, 2002, pp. 41-47.

[7] H. Chan, A. Perrig, and D. Song, "Random Key Predistribution Schemes for Sensor Networks," in IEEE Symposium on Security and Privacy, Berkeley, CA, 2003, pp. 1-17.

[8] W. Du, J. Deng, Y. S. Han, S. Chen, and P. Varshney, "A Pairwise Key Predistribution Scheme for Wireless Sensor Networks," in 10th ACM Conf. Computer and Communications Security, Washington, DC, 2003, pp. 42-51.

[9] D. Liu and P. Ning, "Establishing Pairwise Keys in Distributed Sensor Networks," in 10th ACM Conf. Computer and Communications Security, Washington, DC, 2003, pp. 52-61.

[10] R. Blom, "An Optimal Class of Symmetric Key Generation," in Conference on the Theory and Applications of Cryptographic Techniques, Paris, Fr, 1984, pp. 231-236.

[11] C. Blundo, A. D. Santis, A. Herzberg, S. Kutten, U. Vaccaro, and M. Yung, "Perfectly-Secure Key Distribution for Dynamic Conferences," in 12th Annual International Cryptology Conference on Advances in Cryptology, Santa Barbara, CA, 1992, pp. 471-486.

[12] X. Hai-tao, "A Cluster-Based Key Management Scheme for MANET," in 3rd Int'l Workshop on Intelligent Systems and Applications, Wuhan, China, 2011, pp. 1-4.

[13] R. PushpalLakshmi, R. Rahul, and A. Kumar, "Mobile Agent Based Composite Key Management Scheme for MANET," in Int'l Conf. on Emerging Trends in Electrical and Computer Technology, Nagercoil, India, 2011, pp. 964-969.

[14] F. Liu, X. Cheng, L. Ma, and K. Xing, "SBK: A Self-configuring Framework for Bootstrapping Keys in Sensor Networks," IEEE Transactions on Mobile Computing, vol. 7, no. 7, July 2008 pp 1-11.

[15] P. Loree, K. Nygard, and X. Du, "Efficient Post-Deployment Key Establishment Scheme for Heterogeneous Sensor Networks," in IEEE GLOBECOM, Honolulu, HI, 2009, pp. 1-6.

[16] P. Loree, "Post-deployment Key Management in Heterogeneous Sensor Networks," North Dakota State University, Fargo, ND, 2010, pp. 1-63.

[17] X. Zhao, Y. Lv, T. H. Yeap, and B. Hou, "A Novel Authentication and Key Agreement Scheme for Wireless Mesh Networks," in IEEE 5th Int'l Joint Conf. on INC, IMS, and IDC, 2009, pp. 471-474.

[18] A. Boukerche, Y. Ren, and S. Samarah, "A Secure Key Management Scheme for Wireless and Mobile Ad Hoc Networks Using Frequency-Based Approach: Proof and Correctness," in IEEE Global Telecommunications Conf., 2008, pp. 1-5.

[19] K. K. Chauhan and S. Tapaswi, "A Secure Key Management System in Group Structured Mobile Ad hoc Networks," in IEEE Int'l Conf. on Wireless Communications, Networking and Information Security, Beijing, China, 2010, pp. 307-311.

[20] L. Lu, Z. Wang, W. Liu, and Y. Wang, "A Certificateless Key Management Scheme in Mobile Ad Hoc Networks," in 7th Int'l Conf. on Wireless Communications, Networking and Mobile Computing, Wuhan, China, 2011, pp. 1-4.

[21] H. Dahshan and J. Irvine, "A Trust Based Threshold Cryptography Key Management for Mobile Ad Hoc Networks," in IEEE 70th Vehicular Technology Conf., 2009, pp. 1-5.

[22] U. Sehgal, K. Kaur, and P. Kumar, "Security in Vehicular Ad-hoc Networks," in 2nd International Conference on Computer and Electrical Engineering, 2009, pp. 485-488.

[23] M. Rabin, "Digitalized Signatures and Public-Key Functions as Intractable as Factorization," Cambridge, MA, 1979, pp. 1-20.

Studies on nVidia GPUs in Parallel Computing for Lattice Quantum Chromodynamics and Computational Fluid Dynamics Applications

Simone Coscetti
Ph.D. School of Engineering, Unversità di Pisa
National Institute for Nuclear Physics (INFN) - Pisa, Italy
e-mail: simone.coscetti@pi.infn.it

***Abstract*—During the last few years, we have had the opportunity to watch a fast evolution in the devices for scientific computing. Looking at the Top500 Supercomputer Sites rank, we can see that the most important sites are building new infrastructures with accelerators devices beside traditional Central Processing Units (CPUs). One of the main reasons of this behavior is the investigation of a way to reduce power consumption. For this purpose nowadays the trend is to reduce the amount of memory over the single core in each computing device. The most promising approach to reach this goal is the utilization of Graphics Processing Units (GPUs) alongside traditional CPUs. In this paper, we show a study on nVidia GPUs in parallel computing applied to two demanding fields, such as the Lattice Quantum ChromoDynamics and the Computational Fluid Dynamics.**

Keywords-Parallel Computing; Graphics Processing Units; Lattice Quantum Chromodynamics; Computational Fluid Dynamics.

I. INTRODUCTION

Over the last 20 years, the computing revolution has created many social benefits. The computing energy and environmental footprint have grown, and as a consequence the energy efficiency is becoming increasingly important. The evolution toward an always-on connectivity is adding demands for efficient computing performances. The result is a strong market that pulls for technologies that improve processor performance while reducing energy use.

The improvements in energy performance have largely come as a side effect of the Moore's law [1] - the number of transistors on a chip doubles about every two years, thanks to an ever-smaller circuitry. An improvement in better performance and in energy efficiency is due to more transistors on a single computer, with less physical distance between them.

In the last few years, however, the energy-related benefits resulting from the Moore's law are slowing down [2][3][4], threatening future advances in computing. This is caused by the reaching of a physical limit in the miniaturization of transistors.

The industry's answers to this problem for now are new processor architectures and power efficient technologies.

For decades, the CPU of a computer has been the one designated to run general programming tasks, excelling at running computing instructions serially, and using a variety of complex techniques and algorithms in order to improve speed.

GPUs are specialized accelerators originally designed for painting millions of pixels simultaneously across a screen, doing this by performing parallel calculation using simpler architecture. In recent years the video game market developments compelled GPUs manufacturers to increase the floating-point calculation performance of their products, by far exceeding the performance of standard CPUs in floating point calculations, as illustrated in Figure 1. The architecture evolved toward programmable many-core chips that are designed to process in parallel massive amounts of data. These developments suggested the possibility of using GPUs in the field of High-Performance Computing (HPC) as low-cost substitutes of more traditional CPU-based architectures: nowadays such possibility is being fully exploited and GPUs represent an ongoing breakthrough for many computationally demanding scientific fields, providing consistent computing resources at relatively low cost, also in terms of power consumption (watts/flops). Due to their many-core architectures, with fast access to the on-board memory, GPUs are ideally suited for numerical tasks allowing for data parallelism, i.e., for Single Instruction Multiple Data

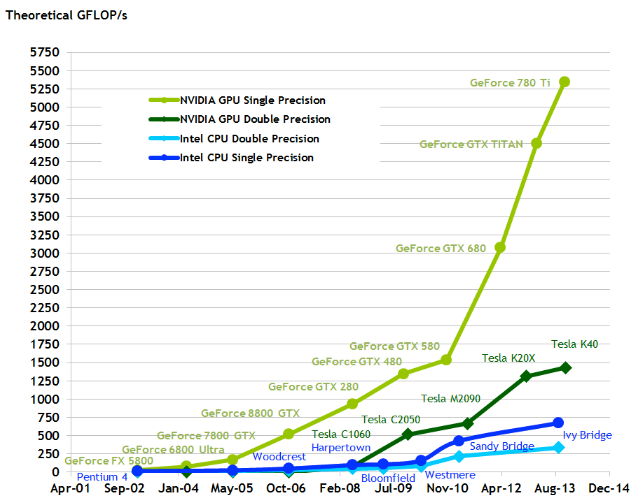


Figure 1. Floating-point operations per seconds for the CPUs and the GPUs [5]

(SIMD) parallelization.

In this work, the parallel computing in Lattice Quantum Chromodynamics (Lattice QCD or LQCD) and in Computing Fluid Dynamics (CFD) using multi-GPUs systems is presented, highlighting the approach of the software in each case, and trying to understand how to build and how to exploit the next generation clusters for scientific computing. In Section 2, the Pisa National Institute for Nuclear Physics (Istituto Nazionale di Fisica Nucleare, INFN) data center is described. The application fields and the computing tools used are introduced in Section 3. In Section 4, the approach to the study and the obtained results are described. Finally, in section 5 the results are discussed.

II. Background

The numerical simulation of the path integral formulation of the Quantum Field Theory discretized on a Euclidean space-time lattice (Lattice QCD), and the numerical resolution of Navier-Stokes partial differential equations using discretization methods in CFD are two typical demanding fields, where parallel computing is applied.

This work has been developed at the Pisa INFN computing center [6][7]. Two of the main communities that work in the computing center are a theoretical physics one, specialized on field theory like QCD, and a mechanical engineers one, working on automotive engineering, that exploits the computing center for CFD computing since 2002. The other main intended use of the computing center is the Tier2 of Compact Muon Solenoid (CMS), one of the two main experiments working at the Large Hadron Collider (LHC) in Geneva.

The computing center is made up of 1,5 PB of storage (up to 90% CMS data) and consists so far of four clusters (over 3300 cores) dedicated to theoretical physics and engineering activities.

III. Application Fields

Lattice QCD and Computational Fluid Dynamics are two of the most promising and demanding fields for GPUs computing. But they are not the only ones. GPUs are used for the computing in several application fields such as: weather and ocean modeling, computational chemistry, biology, bioinformatics, metagenomic data analysis, earth and space science, computational finance.

However, these particular application fields have been chosen because they are extremely interesting as typical examples of different application contests.

Lattice QCD simulations enable us to investigate aspects of the QCD physics that would be impossible to systematically investigate in perturbation theory. The computation time for Lattice QCD simulations is a strong limiting factor, bounding for example the usable lattice size. Fortunately enough, the most time consuming kernels of the Lattice QCD algorithms are embarrassingly parallel, so today, in a quest to tackle larger and larger lattices, computations are commonly performed using large computer clusters. Moreover, the use of accelerators, such as GPUs, has been successfully explored for several years [8]. More generally, massively parallel machines based on heterogeneous nodes combining traditional powerful multicore CPUs with energy-efficient and fast accelerators are ideal targets for Lattice QCD simulations and are indeed commonly used. Programming these heterogeneous systems can be cumbersome, mainly because of the lack of standard programming frameworks for accelerator-based machines. In most of the cases, reasonable efficiency requires that the code is re-written targeting a specific accelerator, using proprietary programming languages, such as Compute Unified Device Architecture (CUDA) for nVidia GPUs.

OpenACC offers a different approach based on directives, allowing to port applications onto hybrid architectures by annotating existing codes with specific "pragma" directives. A perspective OpenACC implementation of a Lattice QCD simulation code would grant its portability across different heterogeneous machines without the need of producing multiple versions using different languages. However, the price to pay for code portability may be in terms of code efficiency.

A. CUDA

CUDA [9] is a parallel computing platform and application programming interface (API) model created by nVidia. It allows software developers to use CUDA-enabled GPUs for general purpose processing. The CUDA platform is a software layer that gives direct access to the GPU's virtual instruction set and parallel computational elements.

The CUDA platform is designed to work with programming languages, such as C, C++ and Fortran. This accessibility makes it easier for a specialist in parallel programming to utilize GPU resources, as opposed to previous Application Program Interface (API) solutions like Direct3D and OpenGL, which required advanced skills in graphics programming. Also, CUDA supports programming frameworks such as OpenCL and OpenACC.

B. OpenACC

OpenACC [10] is a programming framework for parallel computing aimed to facilitate code development on heterogeneous computing systems, and in particular to simplify porting of existing codes. Its support for different architectures relies on compilers; although at this stage the few available ones target mainly GPU devices, thanks to the OpenACC generality the same code can be compiled for different architectures when the corresponding compilers and run-time supports become available.

OpenACC, like OpenCL, provides a widely applicable abstraction of actual hardware, making it possible to run the same code across different architectures. Contrary to OpenCL, where specific functions (called kernels) have to be explicitly programmed to run in a parallel fashion (e.g., as GPU threads), OpenACC is based on pragma directives that help the compiler identify those parts of the source code that can be implemented as parallel functions. Following pragma instructions the compiler generates one or more kernel functions – in the OpenCL sense – that run in parallel as a set of threads.

Regular C/C++ or Fortran code, already developed and tested on traditional CPU architectures, can be annotated with OpenACC pragma directives (e.g., parallel or kernels clauses) to instruct the compiler to transform loop iterations into distinct threads, belonging to one or more functions to run on an accelerator. Various directives are available, allowing fine-tuning of the application.

IV. APPROACH, METHODS AND RESULTS

In the following, the GPU computing approach in both Lattice QCD and in CFD fields are presented, and the results concerning performance comparisons are shown.

A. *Lattice QCD*

From the Lattice QCD side the existing code [11] has been ported and recoded, writing two versions of the code: one in CUDA and one using OpenACC directives. We used a single portion of the original code used for the entirely simulation. In particular, we coded (using plain C) a discretized version of the Dirac matrix, which included two functions D_{eo} and D_{oe}. OpenACC directives instruct the compiler to generate one GPU kernel function for each of them. The function bodies of the OpenACC version strongly resemble the corresponding CUDA kernel bodies, trying to ensure a fair comparison between codes, which perform the same operations. For both the CUDA and OpenACC versions, each GPU thread is associated to a single lattice site. Data structures are allocated in memory following the Structure of Arrays (SoA) layout to obtain better memory coalescing for both vectors and matrices. The basic data element of both the structures is the standard C99 double complex.

We prepared a benchmark code able to repeatedly call the D_{eo} and the D_{oe} functions, one after the other, using the OpenACC implementation or the CUDA one. The two implementations were compiled respectively with the Portland Group (PGI) compiler, version 14.6, and the nVidia nvcc CUDA compiler, version 6.0.

The benchmark code [12] was run on a 32^4 lattice, using an nVidia K20m GPU; results are shown in Table 1, where we list the sum of the execution times of the D_{eo} and D_{oe} operations in nanoseconds per lattice site, for different choices of thread block sizes. All the computations were performed using double precision floating point values.

TABLE 1. EXECUTION TIME PER LATTICE SIZE FOR THE CUDA AND THE OPENACC IMPLEMENTATIONS.

	D_{eo} + D_{oe} functions	
Block size	***CUDA (ns)***	***OpenACC (ns)***
8,8,8	7.58	9.29
16,1,1	8.43	16.16
16,2,1	7.68	9.92
16,4,1	7.76	9.96
16,8,1	7.75	10.11
16,16,1	7.64	10.46

Execution times have a very mild dependence on the block size and for the OpenACC implementation are in general slightly higher; if one considers the best thread block sizes both for CUDA and OpenACC, the latter is $\simeq$ 23% slower.

A slight performance loss with respect to CUDA is expected, given the higher level of the OpenACC language. In this respect, our results are very satisfactory, given the lower programming efforts needed to use OpenACC and the increased code maintainability given by the possibility to run the same code on CPUs or GPUs, by simply disabling or enabling pragma directives. Moreover, OpenACC code performance is expected to slightly improve in the future also due to the rapid development of OpenACC compilers, which at the moment are yet in their early days.

The development of a complete Lattice QCD simulation code fully based on OpenACC is now in progress.

B. *CFD*

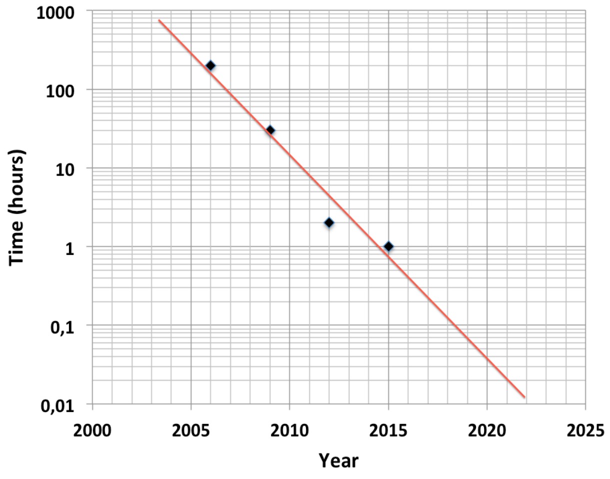
The CFD community that works at the Pisa INFN data center has experienced (from 2002) a reduction of an order of magnitude in the calculation time every 4 years [13]. This huge performance improvement is due basically to the software evolution, and only in small part to the hardware improvement.

During the same period, the number of manageable cells is improved too, from 5 millions in 2002, to 40 millions today. In this case, the hardware is mainly responsible.

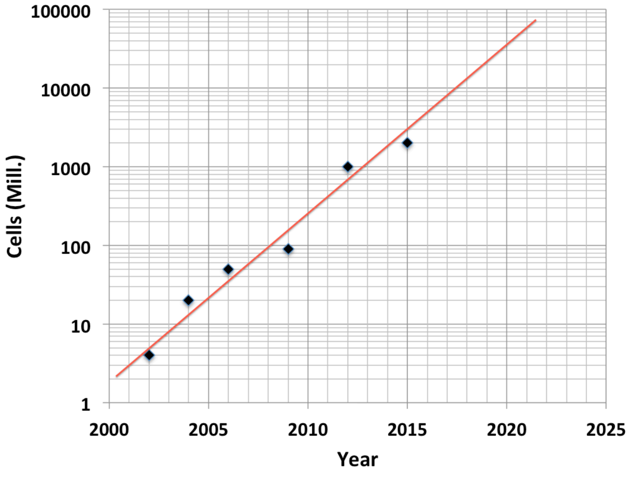
Assuming that the actual trend continues in the next years, the expected scenario is shown in Figure 2.

The ability of simulating 40 millions cells lattice in 1 minute of computation time and managing even 20 billions cells is not trivial. The processors architecture will have to evolve to maintain the trend. The implication will be that both the software and the methodological approach need to evolve and update.

The most promising approach is represented by the employment of GPUs in the simulation chain. However, this technology isn't mature yet, but a version of ANSYS Fluent software enabled to exploit GPUs computing has



(a)



(b)

Figure 2. (a) Computing time and (b) number of cells analyzed during past years with an estimation of the performances in the next years.

been released in late 2013.

One of the main aspects to be considered is that the software behaves differently if GPUs are involved in the computing. In this case, there is an "agglomeration" step on the CPU before the data are sent to the GPUs, and naturally there is a backward step once data have been processed by the GPUs [14][15].

This procedure requires additional time to be performed, and it has two main effects: i) the total execution time will not necessarily be reduced by the exploitation of the GPUs; ii) the performance improvement is difficult to evaluate, taking into account the case and the machine architecture.

The performance improvement depends substantially on the fraction of the whole computation time dedicated to the "Linear Solver". More generally, using only CPUs remains convenient in those cases where CPUs have huge loads (high number of cells per core) and when every single iteration is expensive (double precision, coupled equations, thermal exchange).

The test case is an 8.3 millions cells one, with density base solver, k-epsilon turbulence model [16], with the equation energy activated and a second order discretization accuracy.

The case has been run on a machine set up in the following way:

- CPU: Intel Xeon® E5-2650 v2 @ 2.60 GHz; 2 Socket x 8 cores; 128 GB memory;
- GPU: 2x Tesla K40m, Kepler GK110B, 2880 CUDA cores @ 745 MHz; 12 GB memory.

In Figure 3, one can see how the performance improvement using GPUs decreases increasing the number of CPU cores involved. The improvement has been totally cancelled using all the 16 cores in the machine. We have to underline that due to the limited computing power of the machine, the adopted test case is a lightweight one, both for the cells number and for the model analyzed.

In the case of two CPU cores the benefit of using two GPUs amount to a satisfying 30%.

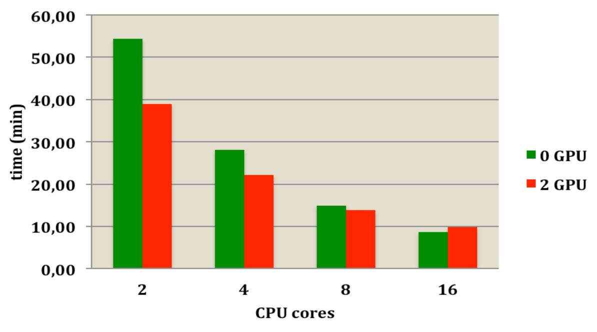


Figure 3. Improvement in AMG with GPUs on a single machine.

V. CONCLUSIONS

Current high performance computing (HPC) systems in the Top500 list have reached petaflops of computational power. Every decade, the computing power of HPC systems increases by a factor of ten. This leads to exascale systems within the next decade. One of the major challenges will be the power. Accelerators offer an unprecedented computational power per watt. Therefore, they are becoming a vital part in todays and future HPC systems. Few dozens systems in the recent (November 2015) Top500 Supercomputing sites list are using either coprocessor or GPGPU technology, including Tianhe-2, Titan, Pin Daint and Stampede in the Top10. Heterogeneous systems combine accelerators and multicore CPU nodes to achieve a better performance while being more energy efficient. One of the major challenges of these systems is the scalability between host CPUs and accelerators.

In order to prepare a mid-size computing center such as the Pisa INFN one for the advent of new architectures dedicated to scientific computing, this work has been finalized to the study of two of the most promising and demanding fields: Lattice Quantum Chromodynamics and Computational Fluid Dynamics. These two fields are extremely interesting as typical examples of different context.

Lattice QCD scientists have a "home-made" code for their simulations, and the approach to a multi-GPU system has been made by porting and recoding the existent code. Different programming approaches, such as CUDA and OpenACC, have been tested, measuring the performances in each case.

Instead, CFD simulations are performed running commercial software (ANSYS Fluent) and the code able to run in a CPUs-GPUs system is not adjustable. In CFD there are two different performance evaluation contexts: how many cells the system can simulate, to more and more defined simulations, and how much time the system spends to perform a simulation with a fixed number of cells, the ideal context for optimizations activities. CFD approach to GPU technology is in its early years, but both hardware and software are proceeding in the right direction.

In particular, for the CFD field, this work represents an in-depth examination whose final target will be the fulfillment of a heterogeneous cluster made of CPUs and GPUs, fully dedicated to the CFD simulations. The purpose is to reach the computing power of 1 PetaFlop, starting from todays 16 TeraFlops cluster (classical CPU-based). The groups concerned to reach the goal will be the CFD-skilled engineers to better understand the simulating strategies, and the ANSYS Fluent experts to configure, set-up and optimize the simulation runs.

GPUs approach is not the only one. The utilization of Xeon Phi coprocessors alongside traditional CPUs has been tested in various environment and application fields and the opinion is that the new generation advent (Knights Landing model) will extend the range of possible applications in fields such as Lattice QCD and CFD.

REFERENCES

[1] G. E. Moore, "Progress in digital integrated electronics", Electron Devices Meeting, vol. 21, 1975, pp. 11-13.

[2] S. Thompson and S. Parthasaranthy, "Moore's law: the future of Si microelectronics", Materials Today, vol. 9, issue 6, 2006, pp. 20-25.

[3] L. B. Kish, "End of Moore's law: thermal (noise) death of integration in micro and nano electronics", Phys. Letters A, vol. 305, issue 3, 2002, pp. 144-149.

[4] R. A. Robison, "Moore's law: predictor and driver of the silicon era", World Neurosurgery, vol. 78, issue 5, 2012, pp. 399-403.

[5] http://docs.nvidia.com/cuda/pdf/CUDA_C_Programming_Guide.pdf [retrieved: February 2016].

[6] S. Arezzini et al., "Optimization of HEP analysis activity using a Tier2 infrastructure", Jou. of Physics: Conf. series, vol. 396, part. 4, 2012, pp. 2054-2059.

[7] S. Arezzini et al., "INFN-Pisa scientific computation environment (GRID, HPC and Interactive Analysis)", Jou. of Physics: Conf. series, vol. 513, track 6, 2014, 062030.

[8] G. I. Egri et al., "Lattice QCD as a video game", Com. Phys. Comm., vol. 177, Issue 8, 2007, pp. 631-639.

[9] https://developer.nvidia.com/about-cuda [retrieved: February, 2016].

[10] http://www.openacc.org [retrieved: February, 2016].

[11] C. Bonati, G. Cossu, M. D'Elia, and P. Incardona, "QCD simulations with staggered fermions on GPUs", Com. Phys. Comm., vol. 183, Issue 4, 2012, pp 853-863.

[12] C. Bonati et al., "Development of scientific software for HPC architectures using OpenACC: the case of LQCD", 2015 IEEE/ACM 1st International Workshop on Software Engineering for High Performance Computing in Science (SE4HPCS), May 2015, pp. 9-15, doi:10.1109/SE4HPCS.2015.9.

[13] A. Ciampa, S. Coscetti, G. Lombardi, and M. Maganzi, "Impact of the power computing breakthrough on the aerodynamic project: an outlook on GPUs utilization in ANSYS-FLUENT", Ansys Convergence, 2015 Regional Conference, 2015, unpublished.

[14] D. Gaudlitz, B. Landmann, and T. Indinger, "Accelerated CFD simulations using Eulerian and Lagrangian methods on GPUs", Procedia Engineering, vol. 61, 2013, pp. 392-397.

[15] V. Sellappan and B. Desam, "Accelerating Ansys Fluent simulations with Nvidia GPUs", Ansys Advantage, vol. IX, issue 1, 2015, pp. 51-53.

[16] B. Mohammadi and O. Pironneau, "Analysis of the k-epsilon turbolence model", 1993.

Benchmarking Big Data Applications: A Review

Sayaka Akioka
School of Interdisciplinary Mathematical Sciences
Meiji University
Tokyo, Japan 164–8525
Email: akioka@meiji.ac.jp

***Abstract*—Big data applications have become one of the non-negligible applications in recent years. These big data applications are supposed to investigate gigantic amount of data from various data sources from several points of view, uncover new findings, and then deliver totally new values. As big data applications handle extremely huge amount of data compared with conventional applications, there is a high, and increasing demand for the computational environment, which accelerates and scales out big data applications. The serious problem here, however, is that the behaviors, or characteristics of big data applications are not clearly defined yet. The appropriate modeling, and benchmarking are indispensable for the development, or design of the adequate computational environment targeting on big data applications. This paper primarily intends to provide a comprehensive survey on modeling, and benchmarking big data applications. We overview, and compare HiBench Benchmark Suite, CloudSuite, and BigDataBench as existing benchmarks for big data applications. We also introduce a couple of projects on modeling big data applications as well.**

***Keywords*—big data; benchmark; stream mining.**

I. Introduction

Big data applications have become one of the non-negligible applications in recent years. These big data applications are supposed to investigate gigantic amount of data from various data sources from several points of view, uncover new findings, and then deliver totally new values. As big data applications handle extremely huge amount of data compared with conventional applications, there is a high, and increasing demand for the computational environment, which accelerates and scales out big data applications. The serious problem here, however, is that the behaviors, or characteristics of big data applications are not clearly defined yet. There is no established model for big data applications right now.

High performance computing community has been investigating data intensive applications, which analyze huge amount of data as well. Raicu et al. pointed out that data intensive applications, and big data applications are fundamentally different from the viewpoint of data access patterns [1]. Therefore, the strategies for speed-up of data intensive applications, and big data applications have to be radically different. Many data intensive applications often reuse input data, and the primary strategy of the speed-up is locating the data close to the target CPUs. Big data applications, however, rarely reuse input data, and this strategy for data intensive applications does not work in many cases. Modern computational environment has been and is evolving mainly for speed-up of benchmarks, such as LINPACK [2], or SPEC [3]. These benchmarks are relatively scalable according to the number of CPUs. Big data applications are not scalable to the contrary, and the current computational environment is not necessarily ideal for big data applications.

This paper primarily intends to provide a comprehensive survey on modeling, and benchmarking big data applications. The appropriate modeling, and benchmarking are indispensable for the development, or design of the adequate computational environment targeting on big data applications. The rest of this paper is organized as follows. Section 2 introduces major benchmark suites for big data applications. Section 3 provides brief review on modeling of a stream mining application, which is one kind of the big data applications. In Section 4, we discuss the current status, and possible future directions on modeling, and benchmarking for big data applications. Section 5 concludes this paper.

II. Benchmark Suite

This section reviews big data benchmark suites in chronological order according to publication dates of the corresponding benchmark suites.

A. HiBench Benchmark Suite

Huang et al. proposed HiBench benchmark suite in 2010 [4]. Huang et al. designed HiBench as a realistic, and comprehensive benchmark suite for Hadoop [5], which is one of the most popular implementations of MapReduce model [6]. HiBench includes both synthetic micro benchmarks, and real-world applications. Before HiBench, GridMix [7], Hive performance benchmarks [8], TeraSort [9], or DFSIO, which is contained in Hadoop source code for benchmarking Hadoop file systems, is the primary option for Hadoop benchmark. Although a benchmark suite is supposed to cover diverse characteristics of target applications, however, Huang et al. pointed out that none of the benchmark suites before HiBench fulfills the requirement for the three reasons;

1) Those benchmark suites require less computations compared to real world Hadoop applications.
2) The data access patterns modeled by those benchmark suites do not assume data access outside MapReduce, such as temporary files on local disks.
3) Some of those benchmark suites focus on more traditional data analysis containing random access using index, or join using partitioning key.

HiBench consists of three micro benchmarks, two Web search related tasks, two machine learning implementations, and one benchmark for Hadoop filesystem (HDFS Benchmark). Table I lists names of the benchmarks. Here, the three micro benchmarks, which are Sort, WordCount, and TeraSort,

TABLE I. CONTENTS OF HIBENCH

Micro Benchmarks	Sort WordCount TeraSort
Web Search	Nutch Indexing PageRank
Machine Learning	Bayesian Classification K-means Clustering
HDFS Benchmark	EnhancedDFSIO

TABLE II. CONTENTS OF CLOUDSUITE

Scale-out Workloads	
Data Serving	Cassandra
MapReduce	Bayesian Classification
Media Streaming	Darwin Streaming Server
SAT Solver	Klee SAT Solver
Web Frontend	a frontend of a web-based social event calendar
Web Search	Nutch
Tradisional Workloads	
CPU	PARSEC, SPEC CINT2006
memory	PARSEC, SPEC CINT2006
Web Frontend	SPECweb09
DBMS	TPC-C, TPC-E
Web Backend	MySQL

are from Hadoop distribution, and these micro benchmarks are widely used for years. In Web search workloads, Nutch Indexing is from Nutch open source search engine [10], and PageRank is a PageRank algorithm implementation [11] from SmartFrog [12]. Two of machine learning workloads are from Mahout [13], which is an open source machine learning library for Hadoop. Bayesian Classification is an implementation of the trainer of Nave Bayes [14]. K-means Clustering is an implementation of K-means clustering algorithm [15]. EnhancedDFSIO captures fine-grained transitions of throughputs of Hadoop file system, while the original DFSIO provides only average numbers.

B. CloudSuite

Ferdman et al. proposed CloudSuite, which is a benchmark suite for big data applications [16]. Table II lists names of workloads included in CloudSuite.

CloudSuite consists of scale-out workloads, and traditional workloads. The scale-out workloads consists of workloads assuming big data applications; Data Serving, MapReduce, Media Streaming, SAT Solver, Web Frontend, and Web Search. Data Serving workload is for NoSQL systems, which is widely utilized popular web applications as the backing store. Cassandra [17] is one of the major implementations of NoSQL. MapReduce workload is for MapReduce model applications, and the implementation is from Bayesian classification from Mahout, which is the same for Bayesian Classification workload in HiBench. Media Streaming workload intends streaming services such as YouTube. Darwin Streaming Server is one of the open media streaming servers [18]. SAT Solver workload represents SAT solvers, which is one of the major supercomputing applications. Klee SAT Solver is a major component of the Cloud9 parallel symbolic execution engine [19]. Web Frontend workload intends the frontend of web applications, which often share the basic functionalities. Web Frontend workload benchmarks the frontend of a web-based social event calendar, and the workload runs Nginx [20] with a built-in PHP module, and APC PHP opcode cache. Web Search workload assumes indexing task of web search engines. CloudSuite includes Nutch as Web Search workload, and this is the same to HiBench Web Search workload. Traditional workloads in CloudSuite consists of well-known benchmarks in each domain, such as PARSEC [21], SPEC [3], and TPC [22].

Ferdman et al. found common characteristics among their workloads;

1) Workloads are scattered across a large number of machines, and each split portion with its data in charge typically fits into the local memory.
2) One workload often consists of a large number of completely independent requests, and these requests do not share any state.
3) Each workload has some special design for cloud computing environment, such as provision for disappearing computational nodes.
4) Inter-machine connectivity is utilized mainly for high-level task management, or coordination.

Through these observations with CloudSuite, Ferdman et al. conjectured that there is a large mismatch between the requirements of big data applications, and predominant processors. They also conjectured that the gap between the requirements, and processor architecture is widening. Here are the conclusions from their study;

1) Big data applications on modern processors suffer from high instruction-cache miss rates.
2) Big data applications have less instruction-level, and memory-level parallelism, and modern out-of-order cores do not provide enough benefits to big data applications.
3) Big data applications handle too huge input data to exceed the capacity of on-chip caches.
4) Big data applications do not require on-chip, and off-chip high bandwidth, which modern processors provide.

Although Ferdman et al. pointed out big data applications behave differently from traditional applications, they did not reveal the reasons for this difference in their work. Therefore, Yasin et al. gave the detailed analysis in [23], and they concluded that big data applications suffer from overheads related to managing the data rather than accessing the data.

Yasin et al. focused on Bayesian Classification from CloudSuite, and they ran the code on Hadoop. Then, they investigated the process from the system level, the application level, and the architecture level (threefold analysis). Their findings are as follows.

1) JVM has a major impact at the system level.
2) There is much room for code optimization at the application level.
3) Hash index lookup is a key limiter, and the optimization at microarchitecture implementation level improves overall performance significantly.
4) Bottlenecks in big data applications are fairly distributed compared to the bottlenecks in traditional applications.

Here, 1) and 2) are notable findings. That is, 1) indicates that careful choice of JVM has a possibility of direct speedup

TABLE III. THE DATASETS BIGDATABENCH INCLUDES.

Datasets	Data Structures
Wikipedia Entries	unstructured
Amazon Movie Reviews	semi-structured
Google Web Graph	unstructured (directed graph)
Facebook Social Graph	unstructured (undirected graph)
E-commerce Transaction Data	structured
ProfSearch Person Resumes	semi-structured

for big data applications. Additionally, 2) represents elimination of inefficient application code in big data applications simply gives a speed up. Actually, [23] demonstrated that one optimization in the main loop gave 50% speedup. Yasin et al. also pointed out that the programming style of big data applications is another obstacle for JVM and CPU exploitation.

C. BigDataBench

Wang et al. proposed BigDataBench in 2014 [24]. Wang et al. advocated BigDataBench provides the better coverage on benchmarking big data applications compared to traditional benchmarks including HiBench, and CloudSuite for the four reasons as follows.

1) BigDataBenchmark includes not only broad application scenarios, but also diverse real-world datasets.
2) BigDataBenchmark is more data centric, and needs to fulfill“ 4V ”(Volume, Variety, Velocity, and Veracity).
3) Workloads in BigDataBenchmark reflect diversity of the real-world big data applications.
4) BigDataBenchmark covers major infrastructures for big data applications including Hadoop, and Hive.

Table III lists real-world datasets, which BigDataBenchmark includes. (Table III is taken from [24], and the latest version has more datasets [25].) Wang et al. argue that these datasets cover diverse of data types, data sources, and application domains. As shown on Table III, While HiBench, and CloudSuite contains one unstructured text dataset respectively, BigDataBench includes one unstructured text dataset, one semi-structured text dataset, two unstructured graph datasets, one structured table dataset, and one semi-structured table dataset. Besides these datasets, BigDataBench also provides Big Data Generator Suite (BDGS), which generates synthetic structured, semi-structured, or unstructured texts, graphs, or tables.

Table IV lists workloads included in BigDataBench. (Table IV is taken from [24], and the latest version has more workloads [25].) BigDataBench workloads are chosen considering combinations of application scenarios (micro benchmarks, basic store operations, relational query, search engine, social network, and e-commerce), application types (online service, real-time analytics, and offline analytics), data types (structured, semi-structured, unstructured), data sources (text, graph, and table), and big data software stacks (Hadoop, Spark, MPI, Cassandra, and more). HiBench targets only on Hadoop, MapReduce, and Hive for big data software stacks. Wang et al. pointed out CloudSuite failed to fulfill the diversity in both datasets, and workloads.

Through experiments with BigDataBench, Wang et al. obtained the three major conclusions as follows.

1) The intensity of Flowting point instructions in BigDataBench is two orders of magnitude lower than the intensity in the traditional benchmarks such as PARSEC, or SPEC. On the other hand, the average ratio of integer instructions to floating point instructions in big data Applications is around two orders of magnitude higher than the average ratio in traditional benchmarks, with the similar intensities of integer instructions in both big data applications, and traditional benchmarks.
2) The volume of input data has significant impact over performance of big data applications.
3) L1 cache MPKI of big data applications are higher than those of traditional applications as pointed out in experiments with CloudSuite (in Section-sec:CloudSuite), while Wang et al. found that L3 caches are effective for the big data applications listed in BigDataBench.

Jia et al. conducted more detailed characterizations with BigDataBench [26]. The summary of their findings is as follows.

1) Software stack, such as Hadoop, Spark, and Cassandra, have much significant impact over the performance of big data applications than the difference of algorithms does. Therefore, software stacks should be included in benchmark suites for big data applications.
2) L3 cache miss rate, instruction fetch stalls, data TLB behaviors, and snoop responses are the four major microarchitectural level metrics to differentiate behaviors of Hadoop-based applications from those of Spark-based applications.
3) Jia et al. employed clustring technique to categorize workloads in BigDataBench, and proposed ways to extract seven workloads as the projection of the original workloads.

III. MODELING BIG DATA APPLICATIONS

We have reviewed major benchmarks for big data applications in Section II. Here, we point out workloads in those benchmarks always require any data storage, no matter traditional database systems, or NoSQL style systems. Actually, there are two kinds of applications in big data applications. One kind is the applications we reviewed in Section II. The other kind is stream mining, which analyzes data in a lined chronological order on the fly (without saving, or revisiting the original data). There is no major benchmark specific for stream minings, and no characteristics of stream minings is unveiled yet. In this section, we briefly overview researches on modeling stream mining algorithms as a first step for the establishment of benchmark suites.

A. Data Access Pattern Analysis

Akioka et al. proposed a stream mining model with the special focus on data dependencies [27]. The figures in this section are borrowed from [27]. Figure 1 illustrates the overall model of stream mining algorithms. In Figure 1, a stream mining algorithm consists of two parts, stream processing part, and query processing part. While the query processing

TABLE IV. THE WORKLOADS BIGDATABENCH INCLUDES.

Application Scenarios	Application Type	Workloads	Data Types	Data Source	Software Stacks
Micro Benchmarks	offline analytics	Sort	unstructured	text	Hadoop, Spark, MPI
		Grep			
		WordCount			
		BFS		graph	
Basic Datastore Operations	online service	Read	semi-structured	table	Hbase, Cassandra, MongoDB, MySQL
		Write			
		Scan			
Relational Query	realtime analytics	Select QUery	structured	table	Impala, MySQL, Hive, Shark
		Aggregate Query			
		Join Query			
Search Engine	online services	Nutch Server	unstructured	text	Hadoop
	offline analytics	Index			
		PageRank		graph	Hadoop, Spark, MPI
Social Network	online services	Olio Server	unstructured	graph	Apache+MySQL
	offline analytics	Kmeans			Hadoop, Spark, MPI
		Connected Components (CC)			
E-commerce	online services	Rubis Server	structured	table	Apache+JBoss+MySQL
	offline analytics	Collaborative Filtering (CF)	semi-structured	text	Hadoop, Spark, MPI
		Naive Bayes			

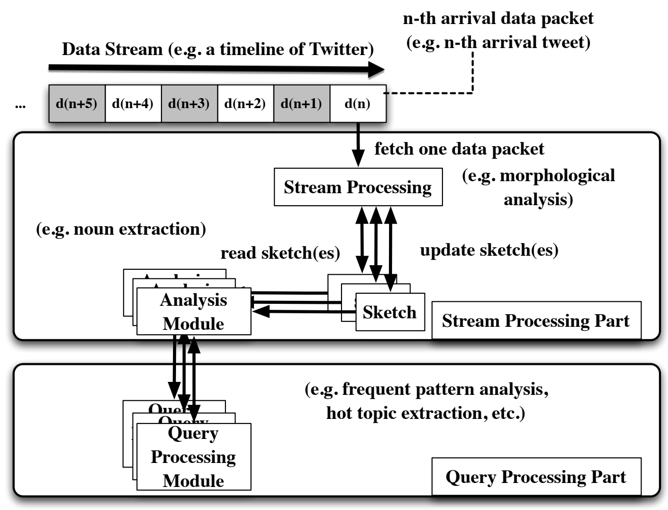


Fig. 1. A model of stream mining algorithms [27].

part is offline analysis, the stream processing part has a huge impact over the performance of the corresponding stream mining application. Once the stream processing part failed to process incoming data on the fly (before the next data unit arrives), the analysis keep losing subsequent data until the stream processing part catches up as there is no buffering space for incoming data in stream mining. In the stream processing part, a sketch is a small memory space similar to a cache. The stream processing module writes the results of preprocessing into the sketch(es), and the analysis module reads from the sketch(es) for the further processing.

Therefore, focusing on stream processing part in Figure 1, Figure 2 illustrates data dependencies between two processes analyzing data units in line, and data dependencies inside the process. The left top flow represents the stream processing part of the precending process, and the right bottom flow represents the stream processing part of the successive process. Each flow consists of six stages; read from sketches, read from input, stream processing, update sketches, read from sketches, and analysis. An arrow represents a control flow, and a dashed arrow represents a data dependencies. In Figure 2, there are three data dependencies in total.

B. Three Layer Model and its Extension

Junghans et al. proposed a three-layer model, which is another model for stream mining algorithm [28]. The figure in this section is borrowed from [28]. The shaded part in Figure 3 illustrates the original three-layer model. The whole picture of Figure 3 illustrates the extended version of three-layer model. The extended part in the model is to optimize the influential parameters in stream mining algorithms for the relaxed resource requirements, or the better quality of the mining results.

The flow of the whole process in the original three-layer model is as follows. First, the filter component filters incoming data stream by sampling, or load shedding. Secondly, the online mining component analyzes the original incoming data stream, or the filtered substream. Thirdly, the results of the online mining component will be stored in the synopsis, which is the second layer of the three-layer model. Here, synopsis indicates sketches, windows, or other dedicated data structures such as a pattern tree. Finally, the offline mining component answers user queries by accessing information stored in the synopsis. Therefore, the offline mining component does not need to fulfill the one pass requirement of stream mining.

The flow in the extension of the three-layer model is as follows. The resource monitoring, and the observation assessment component collect information about the current system state. Based on the monitoring by the resource monitoring, and the observation assessment, the parameters are decided whether they should be updated, or not. Then, the new parameters are set, and the stream mining algorithm run with the updated parameters.

C. Multiple Data Streams

Wu et al. advocated that the existing studies on stream mining assume only one data stream, and proposed formal definition of mining multiple data streams for the better practicality [29]. Wu et al. defined multiple data streams as a set of data flows generated by corresponding sources, and satisfy the following properties:

- continuous, one-pass, sequential, and self-functional,
- distributed, asynchronous, and non-blocking, and

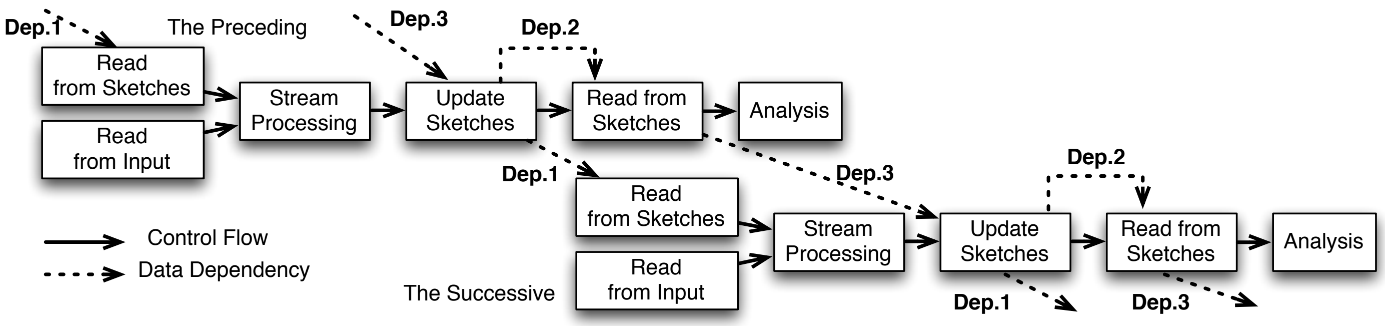


Fig. 2. Data dependencies of the stream processing part in two processes in line [27].

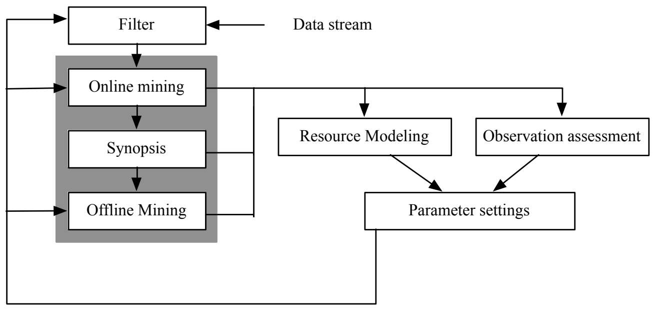


Fig. 3. Extended three-layer model [28].

- diversified, and autonomous.

According to [29], multiple data stream mining should be approached in a separate way from the way for single data stream mining for the following three reasons.

1) Multiple data streams are from many local data sources independently, and these data sources are not be able to process more than simple data preconditioning, or to save all the generated data either.
2) Multiple data stream mining is supposed to analyze across more than one data stream, instead of one single data stream.
3) Multiple data streams are not appropriate to be modeled as one single huge data stream with different attributes as each data is not under uniform timestamp cariteria, sampling rate, or privacy control policy.

In the multiple data stream model, a quadruple of the form (s, t, f, v) represents each data in a data flow, where s is the identification of the place, t is the time, or sequence number identifying the event, f is a function over the data, and v is a value vector of the output. An event refers whether data generation, or any other data processing. Each flow is a set of the quadruples, and fulfills the following properties.

- Each source specifies a single function to generate a single flow.
- If any pair of events, e_1 and e_2, occur at the same source, these two events have the same function invocation, and then the value t of e_1 is smaller than the value t of e_2, e_1 is identified as the event which occured before $e2$.
- For any pair of events, e_1 and e_2, that occur at the different sources, there is no function, or rule between e_1, and $e2$.

Similarly, flows can have some additional properties as follows.

- Homogeneous or heterogeneous: a pair of flows is said to be homegeneous (or heterogeneous) if the respective sources at which the two flows generate specify the same (or different) function(s), which are checked in terms of initial conditions, and output domain.
- Relational: a pair of flows, indicated by f_1, and f_2, is said to be relational if the value vectors of f_1 and value vectors of f_2 satisfy some relationship r.

IV. DISCUSSION

Primarily, the motivation for this paper is to get a quick overview of the current status of benchmarking, and modeling of big data applications. We reviewed several benchmark suites for a certain type of big data applications, and then we also overviewed several studies on modeling of another type of big data applications, which those current benchmark suites do not cover yet. Here are some points we have learned through the survey:

- No single project succeeded to establish a solid model, or solid models for big data applications yet. No single domain of big data applications is clearly characterized, either.
- Although each project has different findings for the details, the common finding is that the current CPU architecture is not suitable for acceleration of big data applications. The speed-up of big data applications seems likely to end up challenging the architecture community.
- Some projects pointed out that some methodologies of software implementations, or some coding styles block speed-up of big data applications. Of course, the way of software development changes according to the trend, or CPU architectures of the moment. The clear thing here is, however, the way of software development, and what CPU architecture community looks at are not meeting well.

- The software layer of big data applications is indispensable, and fat, and then this software layer is another obstacle for clarification of the behaviors of big data applications.

V. CONCLUSIONS

This paper surveyed the major benchmark suites for big data applications, and some projects on modeling of stream mining applications. Although there are several benchmark suites for big data applications as we reviewed, the community have not got the established benchmark suite(s), or model(s) yet. For the serious speed-up of big data applications, CPU artchitecture community, and big data application community move closer to share what each community is looking at.

ACKNOWLEDGMENT

This work was supported by JSPS KAKENHI, Grant Number 15K21423, and 90308271.

REFERENCES

[1] I. Raicu, I. T. Foster, Y. Zhao, P. Little, C. M. Moretti, A. Chaudhary, and D. Thain, "The quest for scalable support of data-intensive workloads in distributed systems," in *Proc. the 18th ACM International Symposium on High Performance Distributed Computing (HPDC'09)*, 2009.

[2] J. Dongarra, J. Bunch, C. Moler, and G. Stewart, *LINPACK Users Guide*, SIAM, 1979.

[3] S. P. E. Corporation. Spec benchmarks. Retrieved: 2016-03-08. [Online]. Available: http://www.spec.org/benchmarks.html

[4] S. Huang, J. Huang, J. Dai, T. Xie, and B. Huang, "The hibench benchmark suite: Characterization of the mapreduce-based data analysis," in *Proc. of 2010 IEEE 26th International Conference on Data Engineering Workshops(ICDEW)*, 2010.

[5] T. A. S. Foundation. Hadoop. Retrieved: 2016-03-08. [Online]. Available: https://hadoop.apache.org

[6] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters," in *Proc. the 6th Conference on Symposium on Operating Systems Design and Implementation (OSDI'04)*, 2014.

[7] T. A. S. Foundation. Gridmix. Retrieved: 2016-03-08. [Online]. Available: https://hadoop.apache.org/docs/r1.2.1/gridmix.html

[8] Y. Jia and Z. Shao. Hive performance benchmarks. Retrieved: 2016-03-08. [Online]. Available: https://issues.apache.org/jira/browse/HIVE-396

[9] T. A. S. Foundation. Package org.apache.hadoop.examples.terasort. Retrieved: 2016-03-08. [Online]. Available: https://hadoop.apache.org/docs/r2.7.1/api/org/apache/hadoop/examples/terasort/package-summary.html

[10] ——. Nutch. Retrieved: 2016-03-08. [Online]. Available: http://nutch.apache.org

[11] P. Castagna. Having fun with pagerank and mapreduce. Retrieved: 2016-03-08. [Online]. Available: http://static.last.fm/johan/huguk-20090414/paolo_castagna-pagerank.pdf

[12] P. Goldsack, J. Guijarro, S. Loughran, A. Coles, A. Farrell, A. Lain, P. Murray, and P. Toft, "The smartfrog configuration management framework," in *ACM SIGOPS Operating Systems Review*, vol. 43, no. 1, 2009.

[13] T. A. S. Foundation. Mahout. Retrieved: 2016-03-08. [Online]. Available: http://static.last.fm/johan/huguk-20090414/paolo_castagna-pagerank.pdf

[14] A. Mahout. Naive bayes. Retrieved: 2016-03-08. [Online]. Available: https://mahout.apache.org/users/classification/bayesian.html

[15] ——. k-means clustering. Retrieved: 2016-03-08. [Online]. Available: https://mahout.apache.org/users/clustering/k-means-clustering.html

[16] M. Ferdman, A. Adileh, O. Kocberber, S. Volos, M. Alisafaee, D. Jevdjic, C. Kaynak, A. D. Popescu, A. Ailamaki, and B. Falsafi, "Clearing the clouds: A study of emerging scale-out workloads on modern hardware," in *Proc. of the 17th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS XVII)*, 2012.

[17] T. A. S. Foundation. cassandra. Retrieved: 2016-03-08. [Online]. Available: http://cassandra.apache.org

[18] M. O. Forge. Darwin streaming server. Retrieved: 2016-03-08. [Online]. Available: http://dss.macosforge.org

[19] D. S. laboratory at EPFL. Cloud9. Retrieved: 2016-03-08. [Online]. Available: http://cloud9.epfl.ch

[20] nginx. nginx. Retrieved: 2016-03-08. [Online]. Available: http://nginx.org/en/

[21] C. Bienia, "Benchmarking modern multiprocessors," Ph.D. dissertation, Princeton University, 2011.

[22] TPC. Tpc benchmarks. Retrieved: 2016-03-08. [Online]. Available: http://www.tpc.org/information/benchmarks.asp

[23] A. Yasin, Y. Ben-Asher, and A. Mendelson, "Deep-dive analysis of the data analytics workloads in cloudsuite," in *Proc. 2014 IEEE International Symposium on Workload Characterization (IISWC2014)*, 2014.

[24] L. Wang, J. Zhan, C. Luo, Y. Zhu, Q. Yang, Y. He, W. Gao, Z. Jia, Y. Shi, S. ZHang, C. Zheng, G. Lu, K. Zhan, X. Li, and B. Qiu, "Bigdatabench: a big data benchmark suite from internet services," in *Proc. 2014 IEEE 20th International Symposium on High Performance Computer Architecture (HPCA)*, 2014.

[25] C. A. o. S. ICT. Bigdatabench—a big data benchmark suite. Retrieved: 2016-03-08. [Online]. Available: http://prof.ict.ac.cn/BigDataBench/#Benchmarks

[26] Z. Jia, J. Zhan, L. Wang, R. Han, S. A. McKee, Q. Yang, C. Luo, and J. Li, "Characterizing and subsetting big data workloads," in *Proc. 2014 IEEE International Symposium on Workload Characterization (IISWC2014)*, 2014.

[27] S. Akioka, H. Yamana, and Y. Muraoka, "Data access pattern analysis on stream mining algorithms for cloud computation," in *Proc. the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA2010)*, 2010.

[28] C. Junghans, M. Karnstedt, and M. Gertz, "Quality-driven resource-adaptive data stream mining," in *ACM SIGKDD Explorations Newsletter*, vol. 13, no. 1, 2011.

[29] W. Wu and L. Gruenwald, "Research issues in mining multiple data streams," in *Proc. the First International Workshop on Novel Data Stream Pattern Mining Techniques (StreamKDD'10)*, 2010.