ADVCOMP 2013

The Seventh International Conference on Advanced Engineering Computing and Applications in Sciences

ISBN: 978-1-61208-290-5

September 29 - October 3, 2013

Porto, Portugal

ADVCOMP 2013 Editors

Sigeru Omatu, Osaka Institute of Technology, Japan

ADVCOMP 2013

Foreword

The Seventh International Conference on Advanced Engineering Computing and Applications in Sciences (ADVCOMP 2013), held between September 29 and October 3, 2013 in Porto, Portugal, continued a series of events that brought together researchers from the academia and practitioners from the industry in order to address fundamentals of advanced scientific computing and specific mechanisms and algorithms for particular sciences. The conference provided a forum where researchers were able to present recent research results and new research problems and directions related to them. The conference included contributions presenting novel research in all aspects of new scientific methods for computing and hybrid methods for computing optimization, as well as advanced algorithms and computational procedures, software and hardware solutions dealing with specific domains of science.

With the advent of high performance computing environments, virtualization, distributed and parallel computing, as well as the increasing memory, storage and computational power, processing particularly complex scientific applications and voluminous data is more affordable. With the current computing software, hardware and distributed platforms effective use of advanced computing techniques is more achievable.

We take here the opportunity to warmly thank all the members of the ADVCOMP 2013 Technical Program Committee, as well as the numerous reviewers. The creation of such a broad and high quality conference program would not have been possible without their involvement. We also kindly thank all the authors who dedicated much of their time and efforts to contribute to ADVCOMP 2013. We truly believe that, thanks to all these efforts, the final conference program consisted of top quality contributions.

Also, this event could not have been a reality without the support of many individuals, organizations, and sponsors. We are grateful to the members of the ADVCOMP 2013 organizing committee for their help in handling the logistics and for their work to make this professional meeting a success.

We hope that ADVCOMP 2013 was a successful international forum for the exchange of ideas and results between academia and industry and for the promotion of progress in the field of advanced engineering computing and applications.

We are convinced that the participants found the event useful and communications very open. We hope that Porto, Portugal, provided a pleasant environment during the conference and everyone saved some time to enjoy the charm of the city.

ADVCOMP 2013 Chairs:

ADVCOMP Advisory Chairs

Chih-Cheng Hung, Southern Polytechnic State University, USA
Juha Röning, Oulu University, Finland
Sigeru Omatu, Osaka Institute of Technology, Japan
Erich Schweighofer, University of Vienna, Austria

ADVCOMP 2013 Research/Industry Chair

Jorge Ejarque Artigas, Barcelona Supercomputing Center (BSC-CNS), Spain
Helmut Reiser, Leibniz Supercomputing Centre (LRZ)-Garching, Germany

ADVCOMP 2013

Committee

ADVCOMP Advisory Chairs

Chih-Cheng Hung, Southern Polytechnic State University, USA
Juha Röning, Oulu University, Finland
Sigeru Omatu, Osaka Institute of Technology, Japan
Erich Schweighofer, University of Vienna, Austria

ADVCOMP 2013 Research/Industry Chair

Jorge Ejarque Artigas, Barcelona Supercomputing Center (BSC-CNS), Spain
Helmut Reiser, Leibniz Supercomputing Centre (LRZ)-Garching, Germany

ADVCOMP 2013 Technical Program Committee

Witold Abramowicz, University of Economics - Poznań, Poland
H. Metin Aktulga, Lawrence Berkeley National Lab, USA
Sónia Maria Almeida da Luz, Polytechnic Institute of Leiria, Portugal / University of Extremadura, Spain
Renato Amorim, University of London- Birkbeck, UK
Gabriel Amorós, Universitat de València, Spain
Sulieman Bani-Ahmad, Al-Balqa Applied University, Jordan
Roberto Beraldi, "La Sapienza" University of Rome, Italy
Simona Bernardi, Centro Universitario de la Defensa / Academia General Militar - Zaragoza, Spain
Mario Marcelo Berón, National University of San Luis, Argentina
Rudolf Berrendorf, Bonn-Rhein-Sieg University, Germany
Ateet Bhalla, Oriental Institute of Science and Technology, India
Muhammad Naufal bin Mansor, University Malaysia Perlis, Malaysia
Pierre Borne, Ecole Centrale de Lille - Villeneuve d'Ascq, France
Kenneth P. Camilleri, University of Malta - Msida, Malta
Juan-Vicente Capella-Hernández, Universitat Politècnica de València, Spain
Yeh-Ching Chung, National Tsing Hua University, Taiwan
Marisa da Silva Maximiano, Escola Superior de Tecnologia e Gestão - Instituto Politécnico de Leiria, Portugal
Vieri del Bianco, Università dell'Insubria, Italy
Javier Diaz, Rutgers University, USA
Juan Carlos Dueñas López, Universidad Politécnica de Madrid, Spain
Jorge Ejarque Artigas, Barcelona Supercomputing Center (BSC-CNS), Spain
Sameh Elnikety, Microsoft Research, USA
Javier Fabra, University of Zaragoza, Spain
Simon G. Fabri, University of Malta - Msida, Malta
Umar Farooq, Amazon.com - Seattle, USA
Mehdi Farshbaf-Sahih-Sorkhabi, Azad University - Tehran / Fanavaran co., Tehran, Iran

Copyright Information

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

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

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

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

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

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

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

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

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

Table of Contents

Semantics and Accuracy of Gene Expression Threshold Computations

A Case Study

Jaime Seguel
Electrical and Computer Engineering Department
University of Puerto Rico at Mayaguez
Mayaguez, Puerto Rico
e-mail: jaime.seguel@upr.edu

Marie Lluberes
Doctoral Program in CISE
University of Puerto Rico at Mayaguez
Mayaguez, Puerto Rico
e-mail: marie.lluberes@upr.edu

Abstract— **The precise inner workings of cellular mechanisms remain largely unknown and, therefore, their modeling is usually based on conjectures. The availability of large amounts of genetic data, and the lack of abstract mathematical models, makes computer algorithms the only tool available for searching for these hypothetical realities. We call the conjectured algorithmic-independent reality that underlies the method design and intention, the semantics of the algorithm. This article is a brief semantics analysis exercise performed with four binary quantization algorithms for time series of gene expression data.**

Keywords- binary quantization; gene expression; quantitative semantics

I. INTRODUCTION

The post genomic era has brought a myriad of computer methods for modeling cellular mechanisms [1]. As the precise inner workings of these mechanisms is still unknown, several of these methods are based on an *implicit model* of the phenomenon under scrutiny, and its hypothetical manifestations in the data. This is a departure from standard scientific computing practices where algorithms are designed on the basis of well-defined mathematical models, which capture the essence of the phenomenon in abstract terms. Explicit mathematical models also guide the design of numerical simulations, and are often used to test their accuracy. The lack of an algorithmic-independent explicit model obscures the essence of the phenomenon simulated by the computer algorithm, as well as the interpretation of its results. Furthermore, taking the implicit model for granted may determine how the phenomenon is perceived and interpreted in further analyses or modeling endeavors [2].

In the absence of a standard term, we call *semantics of a computer method* the algorithmic-independent meaning of the problem that the method is designed to solve. In particular, we study the semantics of four binary quantization algorithms designed to separate expressed from non-expressed gene states, in a time series of gene expression measurements. Underlying the selected methods is the implicit model of a numeric value, or *threshold,* which separates the state expressions of the gene, in the time interval of the series. Such separation is called *binary quantization*, and the algorithms for splitting the time series data points in expressed and non-expressed states, *binary quantization algorithms*.

Two different ways of computing the threshold are manifested in the four methods. For two of the methods, the threshold is in fact, an explicit numerical value computed before the classification of the data. The other two methods do not compute a threshold, explicitly. Instead, they use statistics to separate the data into expressed and non-expressed states. The threshold is thus, a consequence of the classification of the data points. The semantic question that arises here is what is the algorithmic-independent nature of the threshold. We assume as a working hypothesis that the threshold is a numerical value and attempt to unveil its independent nature through computational experiments performed with the four binary quantization algorithms. The experiments assess the degree of consistency between the computed results and the effects of threshold variations in the simulation of gene regulatory networks (GRN) [3]. The ultimate purpose of a binary quantization is the construction of a probabilistic Boolean network representation (PBN) of a GRN [4]. A PBN is normally derived from prior knowledge of gene interconnections and statistical analyses performed on an array representation of the quantized gene expressions, usually called *binary expression matrix*. Most binary quantization methods are validated on the basis of the quality of the PBN representations derived from them. In order to eliminate the influence of the prior knowledge embedded in the PBN representation, we measure the variations in the binary expression matrices themselves.

The rest of this article is organized as follows: Section II discusses the concept of threshold. Section III is a brief summary of the binary quantization algorithms under consideration. Section IV describes the experiments conducted; and Section V analyses their results. Finally, Section VI summarizes some conclusions of the study.

II. NATURAL THRESHOLD, CONVERGENCE THRESHOLD AND COMPUTATION

Gene expressions are the result of a cascade of processes that are stochastic in nature. However, by the Law of Large Numbers, a smooth non-negative real valued function can approximate the average expression behavior of a significantly large number of cells, in an interval of time [5]. Provided that the variations in this function are large enough,

 ISBN: 978-1-61208-290-5

the lowest and highest values can be associated with expressed and non-expressed gene states, respectively. Hypothetically, someplace within the function range, there should be the point in which Nature separates the two states. We call this hypothetical point *Natural Threshold* (NT). We plan to answer the semantics question by investigating to what extent the result of the methods reveal a NT.

In all the methods considered, the threshold varies with the number of input data points. In order to bound these variations, we compute the *Convergence Threshold* (CT). This threshold is obtained by iterative refinements of the method's thresholds, up until the values fall within a predetermined error tolerance. The data for the iterative refinements is taken from a cubic spline interpolation of the input time series.

The CT is also used to assess the accuracy of the methods. We do this by comparing the method's CT with the threshold of the original time series. We also compare the binary expression matrices derived from these thresholds.

III. THRESHOLD COMPUTATION METHODS

We classify the four methods considered in this study [6], [7], [8], and [9], according to their approach to the search for a threshold. This renders what we call *jump-based* and *cluster-based* methods.

A. Jump-based Methods

It is a frequent practice to use data variations —or *jumps*—, between data points, as a reference for the determination of a threshold value. This approach is taken in [6] and [7]. We refer to these methods as *Algorithm* 1 and *Algorithm* 2, respectively. *Algorithm* 1 computes first the average jump of a sorted version of the input data set, and sets the smallest point in the sorted data that exceeds this value, as the threshold. Fig. 1 shows the pseudo code of *Algorithm* 1.

Algorithm 1. Binarize: INPUT G_i, OUTPUT B_i
$S_i \leftarrow \text{sort}(G_{i,1}, \ldots, G_{i,k})$
for j=1 to k-1 **do**
    $D_{i,j} \leftarrow (S_{i,j+1} - S_{i,j})$
endfor
    $t \leftarrow (S_{i,k} - S_{i,1})/(k-1)$
    $m = \min\{j: D_{i,j} > t\}$
for j=1 to k **do**
    if $G_{i,j} \geq S_{i,m+1}$ **then**
        $B_{i,j} \leftarrow 1$
    else
        $B_{i,j} \leftarrow 0$
    endif
endfor

Figure 1. Algorithm 1. Binarize.

In this description, each $G_i = (G_{i,1}, \ldots, G_{i,k})$ is the k-point time series expression of the *i-th* gene, in a gene set. Thus, variable i is fixed in the routine, but runs in the main program. The main program, in turn, calls *Algorithm* 1 and receives its output B_{ij}; which is the *i-th* row in the binary expression matrix. For our purposes, however, the output is $S_{i,m+1}$, as this is the value that separates expressed and non-expressed states.

Algorithm 2, in turn, uses a multi-scale approach for detecting different jumps at different resolution levels. The method scores the jumps before deciding which one is the threshold. An a-posteriori analysis assesses the reliability of this choice. The algorithm consists of several processes. In general, the method finds step functions with different number of steps, which are also the best approximations to the sorted version of the input data. At each approach, the point where the highest jump occurs is identified for further analysis. A relation between the highest jump and the approximation error incurred by the step function is then computed. A high value for this ratio indicates a strong discontinuity in the sorted data, and therefore, a potential threshold candidate. Step functions are computed with a dynamic programming algorithm that returns a sequence of step functions of minimal Euclidian distance to the original data. With each step function approximation, a cost and break point index is calculated and stored. The cost of a step of the function is the distance to the mean of the approached data segment. The cost of the step function, in turn, is the sum of the costs of its steps. Both, the costs and break point indices are computed using the algorithm whose pseudo code is presented below. As in *Algorithm* 1, the first step is sorting the points in the time series. Fig. 2 shows the pseudo code of *Algorithm* 2.

Algorithm 2. Calculation of optimal step functions
Initialization:
$C_i(0) = c_{iN}$, $i = 1, \ldots, N$
Iteration:
for j = 1 to N -2 **do**
    for i =1 to N - j **do**
        $C_i(j) \leftarrow \min_{d=i \ldots N-j}(c_{id} + C_{d+1}(j-1))$
        $Ind_i(j) \leftarrow \text{argmin}_{d=i \ldots N-j}(c_{id} + C_{d+1}(j-1))$
    endfor
endfor

Figure 2. Algorithm 2. Optimal step functions.

Algorithm 3, shown in Fig. 3, reconstructs the break points from the array *Ind,* computed with the *Algorithm* 2.

Algorithm 3. Compute the break points of all optimal step functions
for j = 1 to N - 2 **do**
    $z = j$
    $P_1(j) = Ind_1(z)$
    if $j > 1$ **then**
        $z \leftarrow z - 1$
        for i = 2 to j **do**
            $P_i(j) \leftarrow Ind_{P_i - 1(j)+1}(z)$
            $z \leftarrow z - 1$
        endfor
    endif
endfor

Figure 3. Algorithm 3. Break points of optimal step functions.

Break points are used to compute the jump size h. The error of approximation e is the Euclidean distance of the step

function to the sorted input data set. The maximum of the radios $q = h/e$, determines the strongest discontinuity.

We refer the reader to [7] for further details on this method.

B. Cluster-based Methods

Cluster-based methods partition the input data set into a predetermined number k of disjoint data subsets, referred as *clusters*. The partition is made on the basis of the nearest center of each cluster. Here, we compare the well-known k-means method, for k =2, and *Algorithm* 4, which is a variant, proposed by the authors, that replaces the mean with the median. We provide no pseudo code for *2-means*, but recall that, in this method, data centers are initialized randomly. In contrast, in *Algorithm* 4, no random initialization is necessary, as shown in its pseudo code in Fig. 4.

Algorithm 4. Median separation. INPUT G_i, OUTPUT T
$S \leftarrow$ sort($G_{i,1}, \ldots, G_{i,m}$)
for j = 1 to m - 1 **do**
    $lm_j \leftarrow$ median ($S_1, \ldots, S_j$)
    $um_j \leftarrow$ median($S_{j+1}, \ldots, S_m$)
    $A_j \mid \leftarrow um_j - lm_j \mid$
endfor
$Ind \leftarrow \text{argmax}_{d=1 \ldots m}(A)$
$lmMax \leftarrow lm_{Ind}$
$umMax \leftarrow um_{Ind}$
$T \leftarrow (umMax + lmMax) / 2$

Figure 4. Algorithm 4. Median Separation.

Algorithm 4 sorts first the m input data points, and for each value j, $1 \leq j \leq m$; computes the median of the first j points and that of the last m – j – 1 data points. Then, it finds the pair of medians that are farther apart and returns their average as the threshold. In both, *Algorithm* 4 and *2-means*, two points, an upper and a lower value, determine the binary quantization. Therefore, if there is a threshold, this is most probably given by their average.

IV. METHODS AND EXPERIMENTS

We designed two experiments. The first uses the threshold computed by each method. The second uses the CT of each method, instead. Both experiments were performed with two different data sets. The first data set [10], which corresponds to the mitotic cell cycle of yeast, is taken from [11]. We took the 17-point time series of four genes, namely cdc24, cdc19, cdc15, and cdc27. The second data set is a 6 x 8 randomly generated matrix of real values between 0 and 1, mimicking an eight point time series of six genes.

All experiments were run in Matlab 7.0.12.635 (R2011a) for Mac.

V. ANALYSIS OF RESULTS

Next is a brief analysis of the experimental results.

A. Numerical Variations of the Thresholds

Fig. 5 shows the thresholds obtained in the first experiment. Algorithms *2* and *2-means* exhibit the closest numerical values, while the thresholds returned by *Algorithm* 1 and *Algorithm* 4 are significantly farther apart. All threshold values are shown in Tables 2 and 3. In order to quantify these observations, we compute the ratio $d_{max}/range$, where d_{max} is the largest distance between thresholds for a given time series, and *range* is the difference between the largest and smallest values in the time series.

The results of these computations are depicted in Fig. 6.

For a more algorithmic-centered classification, we define the *distance between methods* as the Euclidian distance between the vectors formed by the thresholds of each time series of genes. Table 1 shows the distances between each pair of methods.

The shortest and largest distances are highlighted, as well. In both experiments and with the cdc data, the distance from *Algorithm* 1 to *2-means* is the largest. In turn, *2-means*

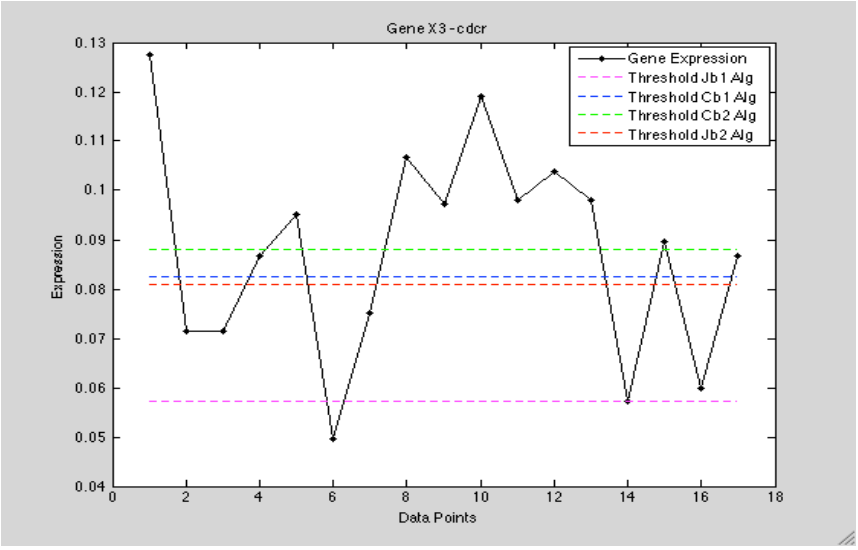


Figure 5. Thresholds plots of the four methods on Experiment 1. Here, Jb1 is Algorithm 1, Jb2 is Algorithm 2, Cb1 is 2-means and Cb2, Algorithm 4.

 ISBN: 978-1-61208-290-5

TABLE I. DISTANCES BETWEEN ALL METHODS EXPRESSED AS HAMMING DISTANCE. LOWEST AND HIGHEST SCORES ARE HIGHLIGHTED.

Hamming Distance							
	Original				*Convergence*		
	Jb2	***Cb1***	***Cb2***		***Jb2***	***Cb1***	***Cb2***
			cdc data				
Jb1	0.23529	0.25	0.32353	***Jb1***	0.45588	0.47059	0.13235
Jb2	—	0.014706	0.088235	***Jb2***	—	0.014706	0.38235
Cb1	—	—	0.073529	***Cb1***	—	—	0.39706
			Random data				
Jb1	0.3125	0.22917	0.3125	***Jb1***	0.4375	0.41667	0.10417
Jb2	—	0.083333	0	***Jb2***	—	0.020833	0.33333
Cb1	—	—	0.083333	***Cb1***	—	—	0.3125

and *Algorithm* 4 are separated by the shortest distance, followed closely by algorithms 1 and 4. We also compared the variations of each algorithm with respect to the input data sets. The results are reported in Table 4. Table 5 shows the Euclidean distance between the two experiments, for each of the four methods. Among the methods, *Algorithm* 1 turned out to be the less stable as it has both the shortest and largest distances between data sets.

B. *Variations in the Binary Quantization*

The Hamming distances divided by the number of data points measure the differences between the binary quantizations derived with each threshold on the same time series. Fig. 7 shows the results for the cdc data set. The highlighted rows are pairs of different methods whose binary quantizations coincide. The methods with the largest number of coincidences are *Algorithm* 2 and *2-means*.

Table 1 shows the Hamming distances between the binary quantization computed with the convergence threshold of each method. The closest methods are again, *Algorithm* 2 and *2-means*. In turn, the distances between algorithms 1 and 4, and algorithms 1 and 2 are the largest.

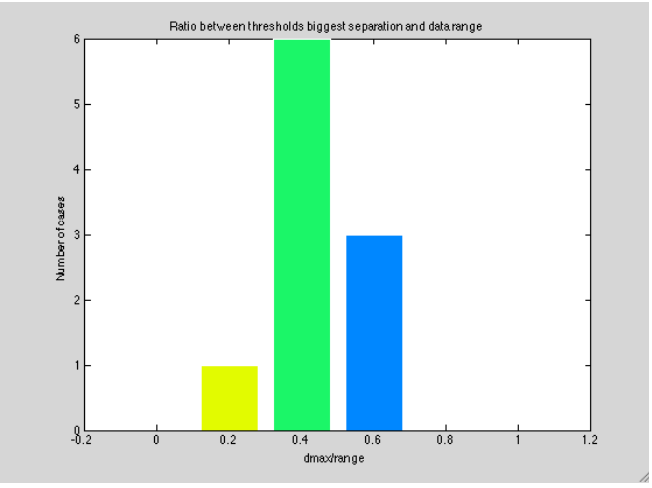


Figure 6. Relations between maximal distance between threshold and data range.

VI. CONCLUSIONS

The results show that the thresholds computed by the four methods are significantly different. This is a clear rejection of the hypothesis that the methods compute the algorithmic-independent value, referred in this article as Natural Threshold. As expected, convergence threshold differs from thresholds. This sensitivity to the sample size is also a negative answer to the question of the accuracy of the methods. Also, the convergence thresholds produced binary expression matrices that are also significantly different to the ones obtained by the thresholds of each method. An important implication that can be drawn from these observations is that the models of gene regulatory networks, whose construction uses a binary quantization as a first step, are biased by the choice of the binary quantization method. The success of some PBN representations of GRNs suggests that this bias is being corrected, in part, with the incorporation of prior gene interconnection knowledge, and expected results.

Original vs. Convergence Binary Quantization Matrices -cdc

Original		Convergence
Jb1		
1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1		1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0		1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1		1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1
0 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 0		0 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 0
Jb2		
0 0 0 0 1 0 1 1 0 1 0 0 1 0 1 0 0 0		1 0 0 0 1 0 1 1 1 1 1 0 0 1 0 1 0 0
0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0		0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
1 0 0 1 1 0 0 1 1 1 1 1 1 1 0 1 0 1		1 0 0 1 1 0 0 1 1 1 1 1 1 1 0 1 0 1
0 1 1 1 1 1 1 1 1 1 1 1 0 1 0 1 1 0		0 1 1 1 1 1 1 1 1 1 1 1 0 1 0 1 1 0
Cb1		
0 0 0 0 1 0 1 1 0 1 0 0 1 0 1 0 0 0		1 0 0 0 1 0 1 1 1 1 1 0 0 1 0 1 0 0
0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0		0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
1 0 0 1 1 0 0 1 1 1 1 1 1 1 0 1 0 1		1 0 0 1 1 0 0 1 1 1 1 1 1 1 0 1 0 1
0 1 1 1 1 1 1 1 1 1 0 0 1 0 1 1 0		0 1 1 1 1 1 1 1 1 1 0 0 1 0 1 1 0
Cb2		
0 0 0 0 1 0 1 0 0 1 0 0 0 0 0 0 0 0		1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0		1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 0 0 0 1 0 0 1 1 1 1 1 1 1 0 1 0 0		1 0 0 0 1 0 0 1 1 1 1 1 1 1 0 1 0 0
0 1 1 1 1 1 1 1 1 1 0 0 1 0 1 1 0		0 1 1 1 1 1 1 1 1 1 0 1 0 1 1 0

Figure 7. Comparison between binary quantization matrices between same methods using thresholds obtained from both experiments. Matching binarizations are highlighted. Cdc data set.

 ISBN: 978-1-61208-290-5

TABLE II. THRESHOLD VALUES FOR RANDOM GENERATED DATA. HIGHLIGHTED VALUES ARE MATCHING VALUES ON BOTH EXPERIMENTS.

Experiment 1				Experiment 2			
Jb1	***Jb2***	***Cb1***	***Cb2***	***Jb1***	***Jb2***	***Cb1***	***Cb2***
0.28072	0.54384	0.50574	0.45214	0.28072	0.48069	0.4671	0.18751
0.2805	0.4284	0.45131	0.41986	0.2805	0.37147	0.40742	0.23781
0.34758	0.45847	0.49741	0.44159	0.34758	0.77806	0.70254	0.37476
0.40338	0.50213	0.51927	0.47716	0.32267	0.50211	0.51097	0.47145
0.51041	0.71765	0.66833	0.64514	0.34708	0.67214	0.64782	0.32805
0.13834	0.60205	0.30395	0.61079	0.13834	0.50511	0.52622	0.15539

TABLE III. THRESHOLD VALUES FOR CDC DATA. HIGHLIGHTED VALUES ARE MATCHING VALUES ON BOTH EXPERIMENTS.

Experiment 1				Experiment 2			
Jb1	***Jb2***	***Cb1***	***Cb2***	***Jb1***	***Jb2***	***Cb1***	***Cb2***
0.1981	0.27476	0.27286	0.30952	0.18095	0.25524	0.25531	0.19169
5.6067	4.1914	3.7892	3.8212	0.85382	3.2098	3.3162	0.89969
0.057143	0.080952	0.082453	0.087976	0.057143	0.080952	0.082453	0.087976
0.11143	0.1181	0.12625	0.125	0.11143	0.11976	0.12571	0.1196

TABLE IV. EUCLIDEAN DISTANCE BETWEEN DIFFERENT METHODS ON SAME EXPERIMENT.

		Jb1—Jb2	***Jb2—Cb1***	***Cb1—Cb2***	***Jb1—Cb1***	***Jb2—Cb2***	***Jb1—Cb2***
Exp. 1	***cdc***	1.41759	0.40229	0.04899	1.81927	0.37196	1.78929
	random	0.60920	0.30835	0.32162	0.40995	0.12134	0.55177
Exp. 2	***cdc***	2.35729	0.10658	2.41736	2.46367	2.31099	0.05689
	random	0.71131	0.09108	0.67506	0.67320	0.71282	0.18449

TABLE V. EUCLIDEAN DISTANCE BETWEEN SAME METHODS ON DIFFERENT EXPERIMENT

		Jb1—Jb1	***Jb2—Jb2***	***Cb1—Cb1***	***Cb2—Cb2***
Exp1 vs Exp2	***cdc***	4.75291	0.98180	0.47333	2.92389
	random	0.18218	0.34761	0.30885	0.64467

The difficulties in determining a numerical threshold may arise from the intrinsic nature of gene expressions. Both assumptions, jumps in the data or statistical separation in two groups may be too strict, in some sense, as data may have some natural perturbation or noise. It may be the case that on average, expressed and not expressed gene states are separated in nature by an interval, not a point. In the interval model, expressed states will correspond to values above the interval's upper limit while non-expressed states, to values below its lower limit. And these expression values that fall within the interval shall be declared *noisy data-points*. Threshold intervals in gene expression time series may be investigated by adding filters that eliminate expression values that are too close to the threshold points returned by the previous methods.

Threshold computation is not a large-scale problem, at least not with the amount of data compilation currently available. However, this may change as models evolve and parameters, such as time, are incorporated. In such cases, parallel and distributed computing versions of the algorithms will be a necessary algorithmic development. Most probably, because of the strong interdependence of data expression, these methods will be mostly implemented in shared memory systems.

This paper suggests a line of research that may be worth pursuing. Its ultimate aim should be a mathematical framework for validating implicit models from their different algorithmic approaches. This validation might eventually lead to, or replace an explicit abstract mathematical representation of the reality behind by the implicit model. The development of such a framework will support current tendencies of using multi-algorithmic approaches to data based computational modeling.

ACKNOWLEDGMENT

This research was supported in part by grants NIH-MARC 5T36GM095335-02 and NIH-R25GM088023 from the National Institute of General Medical Sciences.

REFERENCES

[1] H. Kitano, "Systems Biology: a brief overview," vol. 295, no. 5560, Science, March 2002, pp. 1662-1664.

[2] Y. Kim, S. Han, S. Choi, and D. Hwang, "Inference of dynamic networks using time-course data," Briefings in Bioinformatics, May 2013, doi:10.1093/bib/bbt028.

[3] C. Needham, I. Manfield, A. Bulpitt, P. Gilmartin, and D. Westhead, "From gene expression to gene regulatory networks in Arabioposis thaliana," BMC Systems Biology, 3:85, Sept. 2009, doi:10.1186/1752-0509-3-85.

[4] I. Shmulevich and E. Dougherty, "Probabilisitic Boolean networks: the modeling and control of gene regulatory networks," SIAM, 2010.

[5] L. B. Klebanov and A.Y. Yakovlev, "A nitty-gritty aspect of correlation and network inference from gene expression data," Biology Direct, vol.3, Aug. 2008.

[6] I. Shmulevich and W. Zhang, "Binary analysis and optimization-based normalization of gene expression data," Bioinformatics, vol. 18, no. 4, April 2002, pp. 555–565.

[7] M. Hopfensitz, et al., "Multiscale Binarization of Gene Expression Data for Reconstructing Boolean Networks," IEEE/ACM transactions on computational biology and bioinformatics, vol. 9, no. 2, March 2011, pp. 487–498.

[8] X. Zhou, X. Wang, and E. R. Dougherty, "Binarization of microarray data on the basis of a mixture model," Molecular cancer therapeutics, vol. 2, no. 7, July 2003, pp. 679–84.

[9] K. Hakamada, T. Hanai, H. Honda, and T. Kobayashi, "A preprocessing method for inferring genetic interaction from gene expression data using Boolean algorithm," Journal of bioscience and bioengineering, vol. 98, no. 6, Jan. 2004, pp. 457–63.

[10] R. J. Cho, et al., "A genome-wide transcriptional analysis of the mitotic cell cycle," Molecular Cell, vol. 2, 1998, pp. 65–73.

[11] http//:arep.med.hsrvsrd.edu/ExpressDB/EDS16/EDS16data.txt.

 ISBN: 978-1-61208-290-5

Simulation of Precipitation in an Aluminum Scandium Alloy using Kinetic Monte Carlo and Density-based Clustering with Noise Algorithms

Alfredo de Moura
IPC – Institute of Polymers and Composites
University of Minho
Guimarães, Portugal
alfredo.moura@gmail.com

Antonio Esteves
Computer Science and Technology Center
Informatics Department
University of Minho
Braga, Portugal
esteves@di.uminho.pt

***Abstract*: The present paper reports the precipitation process of Al_3Sc structures in an aluminum scandium alloy, which has been simulated with a kinetic Monte Carlo (kMC) method. The kMC implementation is based on the vacancy diffusion mechanism. To filter the raw data generated by the kMC simulation, the density-based clustering with noise (DBSCAN) method was employed. kMC and DBSCAN algorithms were implemented in the C language. The undertaken simulations were conducted in the SeARCH cluster at the University of Minho. The study covers temperatures, concentrations, and dimensions, ranging from 578K to 873K, 0.25% to 5%, and 50x50x50 to 100x100x100. The Al_3Sc precipitation was successfully simulated at the atomistic scale. DBSCAN revealed to be a valorous aid to identify the precipitates. The achieved results are in good agreement with those reported in the literature, but we went deeper in the evaluation of the influence of all the simulation and analysis parameters. A parallel version of the kMC algorithm using OpenMP was evaluated, which has not proved advantageous compared to the optimized sequential implementation.**

Keywords - Al_3Sc precipitation; kinetic Monte Carlo; cluster analysis; DBSCAN; OpenMP.

I. INTRODUCTION

Precipitate structures play a fundamental role in the material science due to the capacity of representing strong obstacles for dislocations movements within the material structure.

This paper focuses on the elaboration and application of mechanical statistics knowledge, namely the kinetic Monte Carlo method [1], on the study and prediction of the phenomenon of precipitation in an aluminum alloy. The alloy under analysis is the aluminum scandium alloy [2]. The work that will be documented inhere tackles subjects such as computational mechanics, mechanical statistics (the kinetic Monte Carlo method), material science, the precipitation phenomenon, the diffusion phenomenon, what influences this phenomenon and how to control it and also predict it, as well as data mining (namely clustering) the vital information.

OpenMP [3] is an API that allows shared memory parallelization on multi-core machines. It is based on compiler directives, library routines and environmental variables. OpenMP uses multithreading and is based on the fork-join model of parallel execution. It is through directives, added by the programmer to the code, that the compiler adds parallelism to an application. Since the most promising parallelization strategy for the kMC algorithm uses shared memory, OpenMP is a natural choice. Only if the OpenMP implementation of the KMC algorithm accelerates the sequential version in a scalable manner, we will try a distributed memory parallelization strategy, such as Message Passing Interface (MPI) [4].

The outcome of the work undertaken is a set of software applications that allows us (i) to perform Monte Carlo (MC) simulations with and without OpenMP, (ii) to analyze the results using the Density Based Spatial Clustering of Applications with Noise (DBSCAN) technique [5], and (iii) to compare the simulation results with the classical nucleation theory. Practical results obtained with these applications are (i) reports about the simulation, the analysis of clusters and precipitates with DBSCAN algorithm, and the application of the classical nucleation theory; (ii) files for 3D visualization of the simulation (at various stages over time); and (iii) files for 3D visualization of the precipitates.

The rest of the paper is organized as it follows. Section II presents the related work. Section III summarizes the theory behind the simulation of precipitation with kinetic Monte Carlo. Section IV describes the implementation of the simulation and cluster analysis. Section V presents the results of the simulation and analysis. Finally, Section VI points out some conclusions and areas for future research.

II. RELATED WORK

As computation extends its capacities increasingly, so has the scientific field of nucleation and precipitation modeling. The process of modeling nucleation and precipitation has been achieved at different scales, each one having its own advantages and disadvantages. It has increased the number of publications and studies related with the subject of modeling the precipitation kinetics at the atomistic level [6]. At the atomistic level, the simulation model includes (i) the individual atoms, which are organized in a lattice, and (ii) the interactions among atoms, represented by the number of atomic bonds and several energies. The main materials subjected to such studies are alloy materials, such as Fe-Cu, Fe-P-C, Fe-Cu-Ni-Si, Al-Cu.

 ISBN: 978-1-61208-290-5

Aluminum alloys have also their share of studies by which we would like to outline and focus on the Al-Sc alloy.

Binkele and Schmauder studied the precipitation in the Fe-Cu binary system via atomistic Monte Carlo simulations [7][8]. The Fe-Cu system has a BCC structure and the percentages of Fe and Cu used were 90% and 10%, respectively. In our work we have simulated the Al-Sc alloy, which has a FCC structure and a lower supersaturation: the percentage of scandium varied in the range of 0.25% to 5%. Under these conditions, the precipitation is more difficult to observe. We believe that the formula we used to calculate the real-time in simulation is more accurate than the one mentioned in [7]. In [7][8] is not presented a comparison between kMC simulation results and Classical Nucleation Theory (CNT) [9][10], as was done in the present work.

Bombac and Kuglar also simulated a Fe-Cu alloy with MC. The simulation was based on a residence time algorithm, used a BCC rigid lattice structure and applied a temperature of 873K. The outputs of their study are only the number of precipitates and their dimension [11]. They did not compare the simulation results with theory, and several parameters that influence simulation results were not evaluated.

The work by Lae et al. documents a study in which cluster dynamics simulation is applied to Al-Sc and Al-Zr alloys. The achieved results are compared with MC simulation results and they found a good agreement between both simulations results [12]. Comparing with our work, kMC is employed in [12] just as a comparison tool and no details are given about the kMC simulations.

Clouet et al. have published studies of atomistic Monte Carlo simulations not just based on a binary Al-Sc alloy but also on ternary systems [13]. The results of the Al-Sc alloy simulations were compared with the classical nucleation theory. The simulation applied a residence time algorithm using an FCC rigid lattice. Our approach was inspired by Clouet et al. [13] but we evaluated the influence of all the parameters involved in simulation: lattice size, temperature, Sc concentration, and the number of MC steps.

Clouet and Soisson have published a summary of recent applications of the atomistic diffusion model and of the kinetic Monte Carlo method [14]. The summary covers homogeneous and heterogeneous precipitation caused by thermal aging as well as phase transformation caused under irradiation. To conclude this publication the authors mention that atomistic kinetic Monte Carlo simulations provide a convenient way to simulate and model precipitation kinetics in alloys.

Monte Carlo simulations have also been used on the study of other phenomena. Grain growth, abnormal grain growth, thin film deposition and growth, sintering for nuclear fuel aging, bubble formation in nuclear fuels are just some of those phenomena [15].

The three main contributions of the present work to the reviewed literature are (i) the exhaustive evaluation of all the parameters involved in kMC simulation, (ii) the application of a robust and automatic clustering technique, and (iii) the attempt of accelerating the simulation through the parallelization of kMC with OpenMP.

III. THEORETICAL BACKGROUND FOR KMC SIMULATION

This section summarizes the theory, as a set of equations, behind the simulation of Al_3Sc precipitation with kinetic Monte Carlo.

Transition rate for an aluminum atom is calculated by (1) [8].

$$\Gamma_{AlV} = \nu_{Al} \exp\left(-\frac{\Delta E_{AlV}}{kT}\right) \quad (1)$$

As for (2), it describes the transition rate for a scandium atom [8].

$$\Gamma_{ScV} = \nu_{Sc} \exp\left(-\frac{\Delta E_{ScV}}{kT}\right) \quad (2)$$

The aluminum activation energy is obtained by (3) [8].

$$\begin{aligned}\Delta E_{AlV} = E_{spAl} - n^{(1)}_{AlAl}\varepsilon^{(1)}_{AlAl} - n^{(1)}_{AlSc}\varepsilon^{(1)}_{AlSc} - n^{(2)}_{AlAl}\varepsilon^{(2)}_{AlAl} \\ -n^{(2)}_{AlSc}\varepsilon^{(2)}_{AlSc} - n^{(1)}_{AlV}\varepsilon^{(1)}_{AlV} - n^{(1)}_{ScV}\varepsilon^{(1)}_{ScV}\end{aligned} \quad (3)$$

Equations (4), (5), and (6) describe relations among the number of bonds and the size of the first and second neighborhoods, for an FCC structure [8].

$$n^{(1)}_{AlAl} + n^{(1)}_{AlSc} = Z_1 - 1 \quad (4)$$

$$n^{(2)}_{AlAl} + n^{(2)}_{AlSc} = Z_2 \quad (5)$$

$$n^{(1)}_{AlV} + n^{(1)}_{ScV} = Z_1 \quad (6)$$

where $n^{(1)}_{AlAl}$ and $n^{(2)}_{AlAl}$ are the number of aluminum-aluminum bonds regarding the first and second neighborhood, respectively. $n^{(1)}_{AlSc}$ and $n^{(2)}_{AlSc}$ are the number of aluminum- scandium bonds regarding the first and second neighborhood. $n^{(1)}_{AlV}$ is the number of aluminum-vacancy bonds and $n^{(1)}_{ScV}$ is the number of scandium-vacancy bonds, regarding the first neighborhood. Z_1 and Z_2 are the size of the first and second neighborhoods, respectively.

The scandium activation energy is obtained by (7) [8].

$$\begin{aligned}\Delta E_{ScV} = E_{spSc} - n^{(1)}_{AlSc}\varepsilon^{(1)}_{AlSc} - n^{(1)}_{ScSc}\varepsilon^{(1)}_{ScSc} - n^{(2)}_{AlSc}\varepsilon^{(2)}_{AlSc} \\ -n^{(2)}_{ScSc}\varepsilon^{(2)}_{ScSc} - n^{(1)}_{AlV}\varepsilon^{(1)}_{AlV} - n^{(1)}_{ScV}\varepsilon^{(1)}_{ScV}\end{aligned} \quad (7)$$

Analogously, (8), (9), and (10) describe the number of scandium-scandium bonds, number of aluminum-scandium bonds, number of aluminum-vacancy bonds, number of scandium-vacancy bonds, regarding the first and second neighborhood [8].

$$n^{(1)}_{ScSc} + n^{(1)}_{AlSc} = Z_1 - 1 \quad (8)$$

$$n_{ScSc}^{(2)} + n_{AlSc}^{(2)} = Z_2 \quad (9)$$

$$n_{AlV}^{(1)} + n_{ScV}^{(1)} = Z_1 \quad (10)$$

As a vacancy site is surrounded by twelve nearest neighbors, twelve jump rates are calculated. They are the jump frequency Γ_1, Γ_2, until, Γ_{12}. In the next step of a kMC algorithm, one of these 12 frequencies is selected, based on their values and on a random number: the vacancy will jump to the position of atom *n* that verifies (11) (Figure 1).

Equation (12) describes the computation of the real time of simulation. It is composed by the averaged residence time, multiplied by a factor that takes into account the difference between the simulated vacancy concentration and the real vacancy concentration. Equation (13), which traduces analytically the graphical data vacancy concentration versus temperature obtained in [16], calculates the real vacancy concentration in this kMC algorithm.

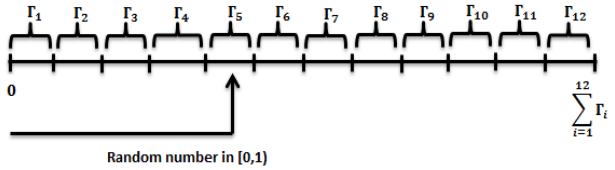


Figure 1. Random selection of the jump frequency.

$$\sum_{i=1}^{n} \Gamma_i \le random\_number \le \sum_{i=1}^{n+1} \Gamma_i \quad (11)$$

$$t_{MC}^{real} = \left(\frac{C_V^{sim}}{C_V^{real}} \right) \times \left(\sum_{i=1}^{12} \Gamma_i \right)^{-1} = \left(\frac{C_V^{sim}}{C_V^{real}} \right) \times t_{MC}^{sim} \quad (12)$$

$$C_{Vreal} = -0.005792301654 + 5.281432466e^{-5} \times T - 1.916781695e^{-7} \times T^2 + 3.466630615e^{-10} \times T^3 - 3.132467044e^{-13} \times T^4 + 1.135950846e^{-16} \times T^5 \quad (13)$$

IV. Implementation of Simulation and Analysis

A. Simulation with Kinetic Monte Carlo

The pseudo-code presented in Figure 2 summarizes the implemented kinetic Monte Carlo algorithm in C language. This code enhances the steps that are of upper importance in a kMC simulation: the activation energy calculation, the vacancy exchange frequency calculation, the step time calculation, the swap of positions between the vacancy and the selected first nearest neighbor. Additionally, the code enhances the step of the data input as well as the step of saving the simulated data.

The correspondent C code is portable, in the sense that it can be compiled and run in any system having `gcc` installed: Linux, Windows or other operating system. As so, the submitted simulations were undertaken in the SeARCH cluster. The SeARCH cluster has the advantage that it can be used to accelerate simulations in three ways: (i) running multiple sequential simulations at same time, with different parameters, (ii) running a parallel simulation on the same machine using OpenMP, or (iii) running a parallel simulation on several machines using MPI. The last option was not implemented since the second alternative was implemented and did not succeed on accelerating the sequential version.

```
main:
  Read the configuration file
  Compute the coordinates of all FCC lattice sites
  Compute average step time and rejection step time →
        → avgStepTime, rejectStepTime
  Initialize the simulated time → timeSim=0
  while (mcs < TOTAL_MCS) do
    Calculate the activation energy → Eact
    Calculate the vacancy exchange frequency and the real time of
    this MCS → vEF, ts
    ts = ts*tsCorrection              // corrected simulated time for
                                      // current MCS
    if (ts > rejectStepTime) then     // step time exceeds a threshold
                                      // that is considered a
                                      // computation error
        Increment errorSteps
        ts = avgStepTime              // replace computed step time by
                                      // average step time
    endIf
    timeSim = timeSim + ts            // accumulated simulated time
    Select a 1st nearest neighbor for the new position of vacancy
    Swap the vacancy with the selected neighbor

    if (mcs = snapshots[numSnap]) then  // if it is a snapshot point
        Save simulation data to VTK | PDB | XYZ file(s)
        snapshotTime[numSnap-1] =timeSim // save snapshot time
        Increment numSnap
    endIf
    Increment mcs
  endWhile
  Write a simulation report to file
end main
```

Figure 2. kMC algorithm.

B. Clustering Analysis with DBSCAN

The main goal of clustering analysis is dividing data into groups, or clusters, which share certain characteristics. Clustering is used in the present work to identify Al_3Sc precipitates in a 3D matrix, containing the position of all Sc atoms, generated by the kMC simulation. The implemented clustering algorithm is designated by DBSCAN [5]. CLARANS [17], DBCLASD [18], and OPTICS [19] are other clustering algorithms, adequate for dealing with large spatial datasets.

As a member of the density-based clustering approaches, DBSCAN identifies regions of high density agglomerations in an immense low density surrounding. Its major advantages are (i) it identifies objects with arbitrary shape and (ii) it does not require that the number of clusters to be identified is provided as input, like *k-means* method does. DBSCAN introduces the notion of noise, used to label atoms that are in low dense regions, which revealed to be an adequate feature in our case. In DBSCAN, for each cluster identified, a point of that cluster is a core point if it has in its neighborhood (with a predefined radius *eps*) a predefined minimum

 ISBN: 978-1-61208-290-5

number of points (*minPts*). DBSCAN classifies points as being: (i) core point - a point in the interior of the density based cluster, (ii) border point - a point that belongs to the border of the density based cluster, and (iii) noise point - a point that is neither a core point nor a border point.

```
DBSCAN (atoms[], nAtoms, eps, minPts)
  cid = 0          // current cluster ID
  pid = 0          // atom position on the array of atoms

  while (pid < nAtoms) do   // cycle over all atoms
    if (atom 'pid' was not  yet visited) then
      Mark atom 'pid' as visited
      Get the size of neighborhood of atom 'pid' → sizeN
      if (sizeN<minPts) then
        Classify atom 'pid' as NOISE
      else
        resBool = ExpandCluster (atoms, nAtoms, visited, N,
                                 pid, cid, eps, minPts)
        if (resBool = TRUE) then
          Increment cid
        endIf
      endIf
    endIf
    Increment pid
  endWhile
end DBSCAN
```

Figure 3. Main function of the DBSCAN algorithm.

```
ExpandCluster (atoms[], nAtoms, visited[], N[], pid, cid, eps,
               minPts)
  Get the size of neighborhood of atom 'pid' → sizeN
  Count unclustered  neighbors of atom 'pid' → sizeUnclustered
  if (sizeUnclustered < minPts) then
    Mark atom 'pid' as NOISE
    return FALSE
  else
    Add atom 'pid' to cluster 'cid'
    for (i in [0:sizeN[) do
      nid = neighbor i-th of atom 'pid'
      if (atom 'nid' was not yet visited) then
        Mark atom 'nid' as visited
        Get size of neighborhood of atom 'nid' → sizeNN
        if (sizeNN  >= minPts) then
          for (j in [0:sizeNN[) do
            nnid = neighbor j-th of atom 'nid'
            Add atom 'nnid' to neighborhood of atom 'pid'
            Increment sizeN
          endFor
        endIf
      endIf
      if (atom 'nid' is not yet member of any cluster) then
        Add atom 'nid' to cluster 'cid'
      endIf
    endFor
  endIf
  return TRUE
end ExpandCluster
```

Figure 4. *ExpandCluster* function used by the DBSCAN algorithm.

The pseudo-code included in Figures 3 and 4 presents the main functionalities of DBSCAN, which was implemented in C language. The code follows the main sequence of steps defined by the authors of the algorithm [5].

To save the atoms belonging to each group was used a data structure that varies dynamically, because the clusters are of variable and unknown size. The used data structure was inspired by the Java *ArrayList* class. After applying DBSCAN, the clusters that are split in several parts are merged in a single spatial region per cluster. This is required because we use periodic boundary conditions (PBC) and aims to improve the 3D visualization of clusters [2].

To permit the visualization of the lattice configurations generated by the kinetic Monte Carlo simulations and by the clustering analysis, these configurations are saved to files in a format that can be read and rendered by available visualization tools. The developed code allows us to save data in one of the following formats: **pdb**, **xyz,** and **vtk**. All these data formats can be visualized with the **ParaView** tool, which is an open-source application adequate to the visualization and analysis of multidimensional data.

Beyond the visualization files with precipitates, the analysis carried out by DBSCAN produces other results, such as, the size and radius of the precipitates, the average size and radius among all precipitates, the percentage of Sc atoms in precipitates and in Al solid solution, and the number of small clusters with the same size.

The main inputs necessary to undergo a simulation and posterior cluster analysis, which are supplied in a configuration file, are: the aluminum lattice constant (Angstrom), the number of unit cells in the x/y/z direction, scandium percentage, simulation Monte Carlo steps, simulation temperature (Kelvin), material parameters, the radius used to define the neighborhood of each atom (***eps*** in DBSCAN algorithm), and minimum number of neighbors that makes an atom to be a core atom of a cluster (***minPts*** in DBSCAN).

The material parameters that supported the previous equations and therefore, the simulations are, first and second nearest-neighbor pair effective energies, saddle point energies and attempt frequencies [13]: $\varepsilon_{AlAl}^{(1)}$ = -0.56 eV; $\varepsilon_{ScSc}^{(1)}$ = -0.65 eV; $\varepsilon_{AlSc}^{(1)}$ =-0.759+21.0x10^{-6}T eV; $\varepsilon_{VV}^{(1)}$ = -0.084 eV; $\varepsilon_{AlSc}^{(2)}$ = 0.113 -33.4x 10^{-6}T eV; $\varepsilon_{AlV}^{(1)}$ =-0.222 eV; $\varepsilon_{ScV}^{(1)}$ = -0.757 eV; e_{Al}^{sp} =-8.219 eV; e_{Sc}^{sp} = -9.434 eV; ν_{Al} = 1.36x10^{14} Hz; ν_{Sc} = 4x10^{15} Hz.

C. Implementation of kMC with OpenMP

Figure 5 presents the algorithm of the main function used to implement the kinetic Monte Carlo simulation with multiple threads of execution, through the OpenMP library. The lines starting with `#pragma omp` specify OpenMP directives, for example to create the parallel threads or to synchronize threads. After the initial steps, which are the same as in the sequential code, it is specified the number of threads to create. The core of the algorithm is a loop that iterates over the number of MC steps. Within this cycle we create parallel threads, each with a private copy of the

specified variables. With the aid of the thread ID (*idT*) and the number of threads (*nT*), each thread can execute only a subset of the calculations.

```
main_OMP

[…] // Initial steps are the same as in non OMP code
// Specify the number of threads to be created
 omp_set_num_threads(numThreads)
Initialize the MC step (mcs) to zero

while (mcs<numberMCStoSimulate) do
   if (idT = 0) then // This section is run by thread with id=0 only
      Count the number of vacancy's first neighbors of Al and Sc type
   endIf

// Create multiple threads
#pragma omp parallel private
   (idT, i, j, nPos, nType, nnPos, nnType, n_AlAl_1, n_AlSc_1,
   n_ScSc_1, n_AlV_1a, n_ScV_1a, n_AlAl_2, n_AlSc_2,
   n_ScSc_2, expoent)
{
   idT = omp_get_thread_num() // ID of each thread
   nT = omp_get_num_threads() // Number of threads
   i = idT
   while (i < NUMBER_1ST_NEIGHBORS) do
      Compute Eact[i] associated with i-th vacancy neighbor
      i = i + nT
   endWhile
   Compute absolute vacancy exchange freq. with its 12  1-st neighbors
   #pragma omp barrier
   if (idT = 0) then
      Compute the sum of all 1-st neighbors absolute exchange freq.
   endIf
   #pragma omp barrier
   Compute relative vacancy exchange freq. with its 1-st neighbors
} // (end of) multiple threads

   if (idT = 0) then
      Sum of all relative vacancy exchange freq. with 1-st neighbors
      totalT = totalT + 1/sumAbsoluteVef
      Select randomly a 1-st nearest neighbor for new vacancy
      Swap the vacancy with the selected neighbor
      if (mcs = snapshots[numSnap]) then
         Save simulation data to file at snapshot
         Increment the number of the current snapshot
      endIf
      Increment the MC step (mcs)
   endIf
endWhile // (end of) cycle relative to the number of MCS

[…] // Final steps are the same as in non OMP code
end main_OMP
```

Figure 5. kMC algorithm with OpenMP.

For example, each thread calculates a subset of the activation energies (*Eact[]*) associated with the 12 neighbors of a vacancy. If it is necessary that all threads reach a certain position in the code, at the same time, it is inserted a synchronization barrier. The condition "*idT=0*" is used to force the calculations to be carried out only by thread 0.

V. RESULTS

Figure 6 illustrates the time evolution of the precipitation phenomenon. The initial random configuration applied to the simulation is shown in Figure 6 (a). The sequence of figures report a simulation that undertook the conditions of 873K, 1%Sc, and over $5x10^{11}$ MCS in a 50x50x50 lattice box ($5x10^5$ atoms). The Sc atoms in raw configurations produced by the simulation are presented in the left part of each figure. The right configuration of each figure demonstrates the application of the DBSCAN algorithm, where the scandium atoms that do not belong to precipitate structures are labeled NOISE and do not appear.

The sequence of graphics from Figure 7 summarizes the analysis undertaken over the simulation outputs. Figure 7 (a) represents the evolution of precipitates dimension in terms of radius measure. Figure 7 (b) acknowledges the evolution of the presence of scandium atoms distributed in the aluminum solid solution. As with Figure 7 (c), it is possible to acknowledge the evolution of the presentage of scandium atoms in precipitate structures. Figure 7 (d) is one of the most important interpretations that is conducted regarding simulation of the nucleation of precipitates as it allows one to undertake comparison analyses with the CNT [9][10]. Two measures are used to efectively compare kMC with CNT: the steady-state nucleation rate (J^{st}), which represents the number of supercritical nuclei formed per unit time in a unit volume and the cluster size distribution (C_{nSc}), which defines the probability to encounter a cluster with a dimension of *n* atoms in a solid solution [2].

The simulations were run on the SeARCH cluster, located at the University of Minho. Table I contains the technical specifications of the SeARCH cluster nodes where we run the kinetic Monte Carlo simulations.

The computation time mainly depends on the number of MC steps. Simulations duration is also influenced by the technical specifications of the machines where the simulations were run. On a `compute-311-X` node of the SeARCH cluster, a simulation with $5x10^{11}$ took around 8 days, and 12 days on a less performing `compute-201-X` node. Computation time does not depend significantly on the scandium percentage, the lattice size or any other parameter of the simulations.

 ISBN: 978-1-61208-290-5

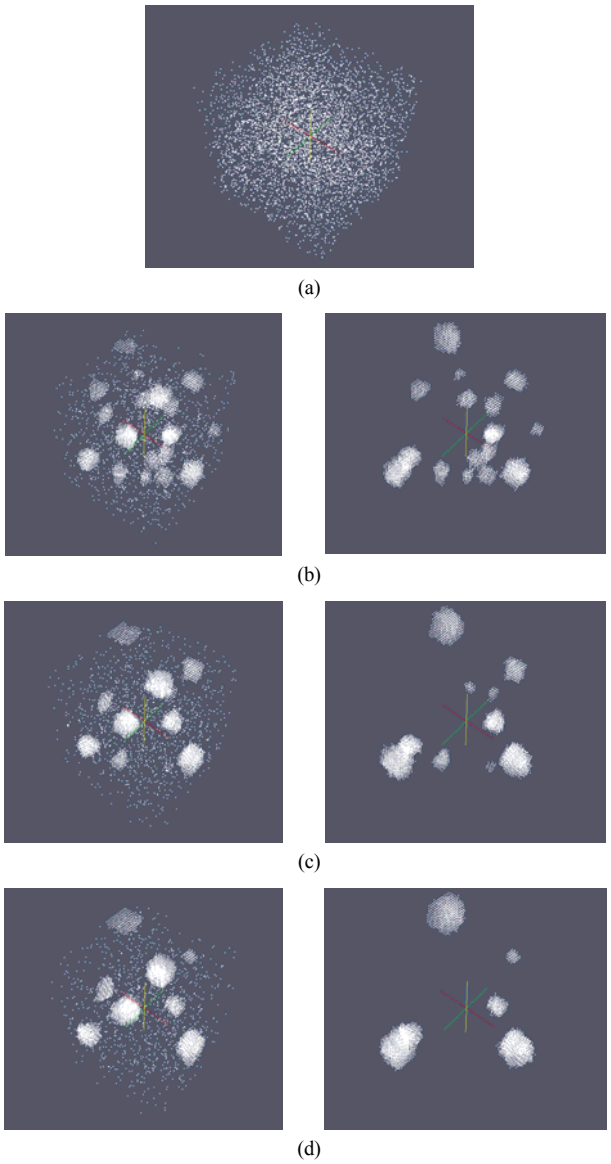


Figure 6. Evolution of simulation: (a) initial configuration; (b) t=1.55ms; (c) t=3.03ms; (d) t=4.945ms (left/right ⇔ before/after applying DBSCAN).

Table II summarizes the computation time needed by a kMC simulation with different number of threads. The number of MC steps simulated was 10^7, the lattice included 10*10*10*4 atoms and we used C code with OpenMP. As we can see from Table II, the utilization of an increasing number of threads is counterproductive. The poor performance achieved by the presented parallel implementation results from 3 facts: (i) the problem we are dealing with is not inherently parallel, since the MC simulation has only one vacancy, (ii) the work assigned to each thread is small and does not compensate the computation overhead introduced by the threads, and (iii) there are several parts of the code that have to be executed by one thread only.

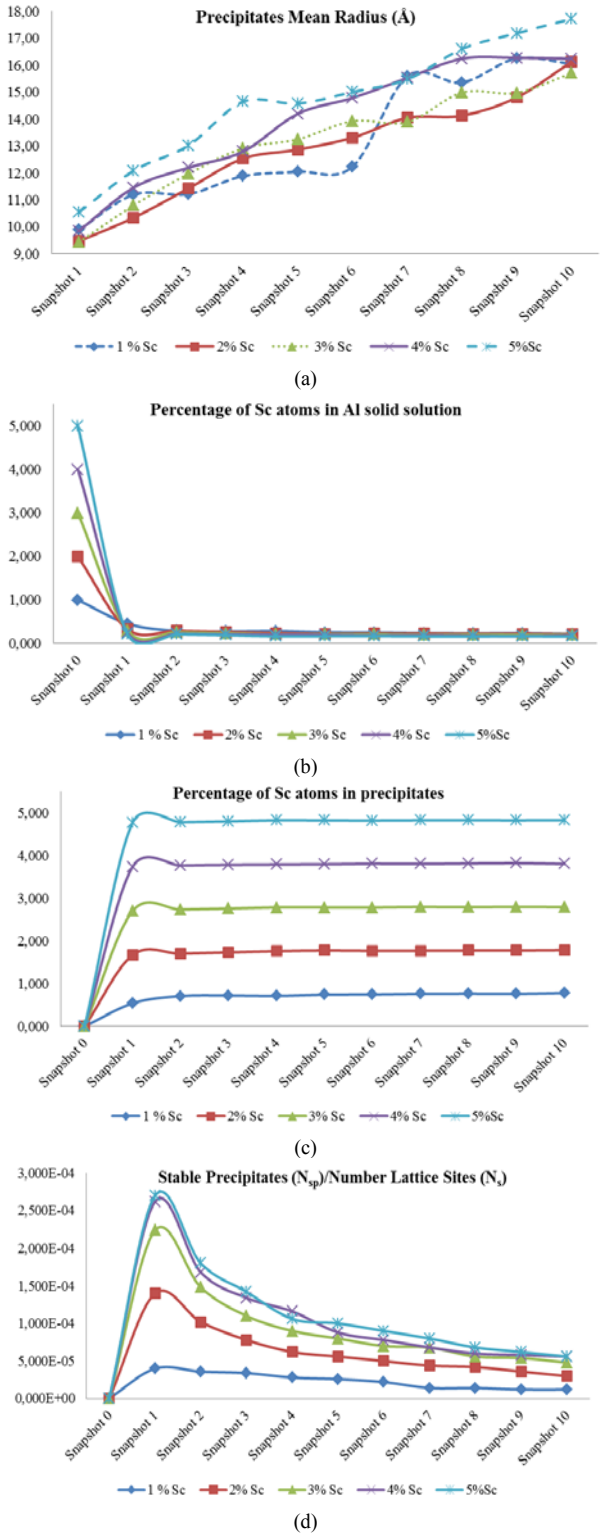


Figure 7. Simulation metrics: (a) precipitates mean radius; (b) percentage of Sc in Al solid solution; (c) percentage of Sc in precipitates; (d) number of precipitates normalized by the number of lattice sites.

 ISBN: 978-1-61208-290-5

TABLE I. TECHNICAL SPECIFICATIONS OF THE SEARCH NODES USED BY THE KMC SIMULATIONS.

Nodes	*Processors*	*CPUs Number*	L2 Cache	*Operating System*
311–X nodes	Intel Xeon E5420	8	12 MB	Linux x86_64
201–X nodes	Intel Xeon 5130	4	4 MB	Linux x86_64
101–X nodes	Intel Xeon	4	2 MB	Linux x86_64

TABLE II. COMPUTATION TIME, NEEDED BY A MC SIMULATION, AS A FUNCTION OF THE NUMBER OF THREADS.

Number of threads	Average computation time (s)
1	25
2	46
4	52
8	62
12	70

VI. CONCLUSIONS AND FUTURE WORK

kMC simulation of Al_3Sc precipitation on a supersaturated Al solid solution was successfully achieved. This proves that the equations used to model Al_3Sc precipitation are correct. The results from kMC simulations were further improved by the application of DBSCAN, which proved to be a valorous aid to identify the Al_3Sc precipitates, by eliminating the unclustered Sc atoms. The DBSCAN algorithm reveals adequate in the role of identifying, visualizing and measuring (size, radius, and shape) of the precipitates embedded in the Monte Carlo output data. By simulating with various Sc percentages, as well as temperatures, the capacity of clustering Al_3Sc precipitates maintains accurate.

The number of stable precipitates strongly increases in the initial phase. After that, the number of precipitates reduces, as predicted by the theory of nucleation. Consequently the surviving precipitates increase in size, either in number of atoms or in radius. The mean precipitates radius increases almost linearly over time. The number of precipitates normalized by the number of lattice sites increases rapidly in the initial phase of the simulation and then decreases slightly during the rest of the simulation. Temperature has a profound influence on the evolution of the precipitation simulation. As the CNT states, and the simulation graphics do prove, the steady state nucleation rate rises with the temperature increase.

The achieved results are very much in good agreement with those reported by Clouet et al. [13]: the increase of the precipitates average size and the reduction of the Sc concentration in the Al solid solution during the simulation follow the same tendency. The comparison between kMC and CNT are very much similar [13]. Although we have used the same model for Al_3Sc precipitation as [13], it was possible to go deeper in the evaluation of the influence of all the parameters involved in simulation: lattice size, temperature, Sc concentration, number of MC steps, and the technique used in cluster identification and measuring. We also tried strategies to accelerate the simulation, using OpenMP.

Some features of ParaView made it an interesting choice for visualization and even analysis such as its support to the three formats (vtk, pdb, xyz) we used as output of kMC, it is open source and based on a popular framework (VTK) [20], and it supports parallelism as to handle huge files.

A field for future research is the exploration of parallelization techniques for the kMC simulation. Due to the sequential nature of the precipitation problem, a hypothesis is to use multiple vacancies and run multiple simulations in parallel, each one with a vacancy and a sub-lattice. Simulating with multiple vacancies alters the vacancy concentration to a unrealistic value. Thus, the validity of this alternative, used to speed up the simulations, has to be demonstrated. Examples of algorithms that follow this strategy are the optimistic synchronous relaxation (OSR) and the semi-rigorous synchronous sub-lattice (SL) [21]. These approaches have to deal with two critical issues: correct the excessive vacancy concentration and synchronize the parallel instances of the asynchronous kMC simulation. Another future research topic would be extending MC method to simulate ternary alloys, such as Al-Mg-Sc, Al-Sc-Si or Al-Sc-Zr.

ACKNOWLEDGMENTS

This work was funded by National Funds through the FCT - Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) within project PEst-OE/EEI/UI0752/2011.

REFERENCES

[1] A. Voter, "Introduction to the Kinetic Monte Carlo Method", Radiation Effects in Solids, Springer, pp. 1-23, 2007.

[2] A. de Moura, “Simulation of the Nucleation of the Precipitate Al3Sc in an Aluminum Scandium Alloy using the Kinetic Monte Carlo Method”, MSc thesis, University of Minho, Dec 2012.

[3] B. Chapman, G. Jost, and R. Pas, "Using OpenMP: Portable Shared Memory Parallel Programming", The MIT Press, 2007.

[4] W. Gropp, E. Lusk, and A. Skjellum, "Using MPI: Portable Parallel Programming with the Message Passing Interface", 2nd edition, The MIT Press, 1999.

[5] M. Ester, H.-P. Kriegel, J. Sanders, and X. Xu, “Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise”, Published in Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining (KDD-96), 1996.

[6] J. Röyset, “Scandium in aluminum alloys overview: physical metallurgy, properties and applications”, Metallurgical Science and Technology, Hydro Aluminium R&D Sunndal, N-6600 Sunndalsöra, Norway.

[7] P. Binkele and S. Schmauder, “An atomistic Monte Carlo simulation of precipitation in a binary system”, International Journal for Materials Research, 94, pp. 1-6, 2003.

[8] S. Schmauder and P. Binkele, “Atomistic computer simulation of the formulation of Cu-precipitates in steels”, Computational Materials Science 24 (2002), 2002, pp. 42-53.

[9] D.Kashchiev, "Nucleation: Basic Theory with Applications", Butterworth-Heinemann, Oxford, 2000.

[10] E. Clouet and M. Nastar, "Classical nucleation theory in ordering alloys precipitating with L12 structure", Physical Review B 75, 132102, pp. 1-4, 2007.

 ISBN: 978-1-61208-290-5

[11] D. Bombac and Goran Kugler, “Precipitation in Alloys: A kinetic Monte Carlo and Class Model Study”, World Journal of Engineering, 2010.

[12] L. Lae, P. Guyot, and C. Sigli, "Cluster dynamics in AlZr and AlSc alloys", Materials Forum Volume 28, Institute of Materials Engineering Australasia Ltd, pp. 281-286, 2004.

[13] E. Clouet, M. Naster, and C. Sigli, “Nucleation of Al_3Zr and Al_3Sc in aluminum alloys: From kinetic Monte Carlo simulations to classical theory”, Physical Review B 69 (6), 064109, pp. 1-14, 2004.

[14] E. Clouet and F. Soisson, “Atomic simulations of diffusional phase transformations”, C. R. Physique 11 (2010), 2010, pp. 266-235.

[15] S. Plimpton, C. Battoile, M. Chandross, L. Holm, A. Thompson, V. Tikare, G. Wagner, E. Webb, and X. Zhou, “Crossing the Mesoscale No-Man´s Land via Paralelel Kinetic Monte Carlo”, Sandia Report, SAND2009-6226, 2009.

[16] J. E. Hatch, “Properties and Physical Metallurgy”, American Society for Metals, 1984.

[17] R. Ng and J. Han, “Efficient and Effective Clustering Methods for Spatial Data Mining”, Proc. 20th Int. Conf. on Very Large Data Bases, Santiago, Chile, pp. 144-155, 1994.

[18] X. Xu, M. Ester, H. Kriegel, and J. Sander, "A Distribution-Based Clustering Algorithm for Mining in Large Spatial Databases", Proceedings of the 14th International Conference on Data Engineering, pp. 324-331, 1998.

[19] M. Ankerst, M. Breunig, H. Kriegel, and J. Sander, "OPTICS: Ordering Points To Identify the Clustering Structure", Proceedings of the ACM SIGMOD International Conference on Management of Data, pp. 49-60, 1999.

[20] W. Schroeder, K. Martin, and B. Lorensen, "The Visualization Toolkit An Object-Oriented Approach To 3D Graphics", 4th Edition, Kitware Inc. publishers, 2006.

[21] G. Nandipati, Y. Shim, J. Amar, A. Karim, A. Kara, T. Rahman, and O. Trushin, “Parallel kinetic Monte Carlo simulations of Ag(111) island coarsening using a large database”, Journal of Physics: Condensed Matter, 21(8):084214, pp.1-12, 2009.

 ISBN: 978-1-61208-290-5

A CADx Scheme in Mammography: Considerations on a Novel Approach

Bruno Matheus, Homero Schiabel
Department of Electrical Engineering
USP – ESSC
São Carlos, Brazil
bmatheus@sc.usp.br, homero@sc.usp.br

***Abstract*—This paper describes a prototype of a complete CADx system developed in the last years in our research groupp. Its basic structure consists of pre-processing corrections based on the image acquisition and digitization procedures (FFDM, CR or film + scanner), a segmentation tool to detect clustered microcalcifications and suspect masses and a classification scheme, which evaluates the presence of microcalcifications clusters and possible malignant nodules based on their contour. The aim is to provide enough information not only on the detected structures but also a pre-report with a BI-RADS classification. At this time, the system is still lacking an interface integrating all the modules. Despite this, it is partially functional, as a prototype for testing.**

Keywords-Mammography; Mammographic CAD; Microcalcification Detection; Nodule Detection; Nodule Classification; Image Analysis.

I. INTRODUCTION

Since all women over the age of 40 are recommended to perform mammographic exams every two years, the demands on radiologists to evaluate mammographic images in short periods of time has increased considerably. As a tool to improve quality and accelerate analysis, CADe/Dx (computer-aided detection/diagnostic) systems are being researched, but very few complete CADe/Dx systems have been developed and most are restricted to detection and not diagnosis. The existent ones [1] [2] are associated to specific mammographic equipment (usually Digital Radiology), which makes them very expensive.

Computer-aided Diagnosis (CAD) schemes are addressed to accelerate and ease the evaluation of medical images, sometimes serving as a second opinion. In mammography, the lack of trained professionals makes the careful evaluation of each image of each exam expensive and restrictively time consuming, especially for second opinions. Thus, several CAD schemes are being developed focusing on mammography. However, the breast tissue is commonly very difficult for analysis, which makes the few CAD schemes approved by the American Food and Drug Administration (FDA) highly expensive, in addition to be only designed for detection (CADe) – indicating suspected signals without classifying them.

Our research group has been developing in the past years some processing schemes for both detection and classification of signals regarding mammographic images. These schemes include special attention to pre-processing enhancements based on imaging acquisition quality characteristics evaluation [3] [4] [5], detection of clustered microcalcifications [6] [7] and detection and classification of suspected masses [8] [9] [10] [11].

The purpose of this paper is to present the ensemble of several years of research in this CADx scheme prototype, including the results obtained individually by each module (as well as joined at all) and to discuss the further developments of this system in the near future. The scheme corresponds to an automatic CADx system being developed for free use and, in the future, to be connected to an online image database [12] to aid in the medical reports.

This following paper contains a Methods section, where the prototype system is described; this section is subdivided by the function of the module described. The next section is Results, where the results obtained so far with the prototype are described. Lastly, the Conclusion section discuss the future of the project.

II. METHODS

This CADx scheme is divided in 3 major structures: Pre-processing, CADx processing and Comparison to database. A detailed diagram is presented in Figure 1. Each division is described as following:

A. Pre-processing

Once the digital images are obtained, they are submitted to pre-processing techniques for enhancement based on imaging intrinsic parameters, determined by quality assurance procedures. This process is composed of:

A.1 Breast and pectoral muscle segmentation

Initially, the digital image is segmented to remove tags and as much as possible of the background, reducing the size of the image and, consequently, reducing the computational cost of the following procedures. In this step, the pectoral muscle is also segmented to be used in step B.B.4.

This method consists of a logarithm contrast enlargement [13] [14] followed by a segmentation using Pun's global threshold [15]. With this, the breast itself is

 ISBN: 978-1-61208-290-5

segmented from the background. The techniques used in this step are fully detailed in [16].

A.2. Scanner/CR corrections

Another source of errors in this process is the image conversion in digital format. The aim in this step is "to correct" the final image according to the errors or distortions due to the digitization process, from a film digitizer or other electronic device in Computed Radiography (CR) or even Digital Radiology (DR) systems [5]. The procedure essentially transforms the image intensity according to the ideal relation between the optical density (or the beam intensity) and pixel gray scale. This is particularly important in film scanning, due to the relation between the film sensitometric curve and the digitizer characteristic transfer curve.

The technique used in this stage was previously described. For its use, knowing the characteristic curve of the scanner or CR/DR system is a requirement.

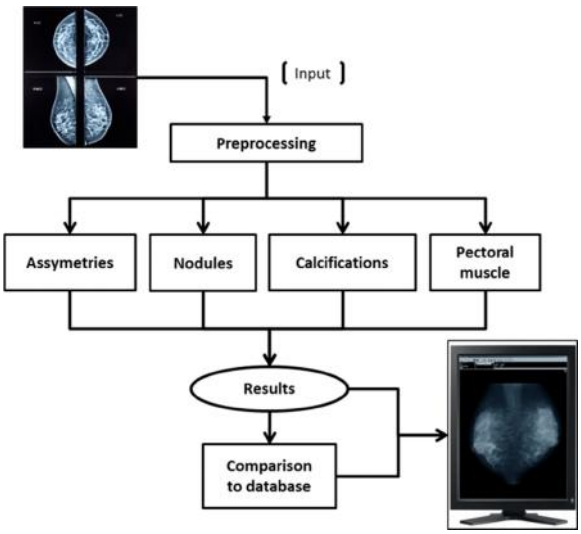


Figure 1. CADx diagram

B. CADx processing

Once the image is digitized and transformed according to the pre-processing techniques, the image is processed in the 4 distinct modules below:

B.1 Density

In this module, the breast image has its density evaluated, according to BIRADS [16] [17] classifications. This evaluation is performed based on the average gray scale intensity for the entire breast. Also, in this module, both breasts are compared in each orientation (Mediolateral-Oblique - MLO and Cranio-Caudal - CC) and analyzed for density differences between breasts, characterizing an asymmetry. If an asymmetry is detected, it is evaluated as a nodule detection in step B.2.

The results of this system are of an average sensitivity of 94.6% with a false-positive rate of 26.35%. Comparing to a well-recognized work in the field [18] - with a sensitivity of 90.38% and 32.12% false-positives rate - the proposed module has better results.

B.2Nodules

This module is designed to detect and classify nodules. First, the complete image is evaluated using neural network techniques to detect Regions of Interest (RoIs), which contain nodules [16]. The detected RoIs are then segmented in order to separate the nodule itself from the background using EICAMM [19]. Once the nodule is segmented, an evaluation of the contour, texture and density is used to classify the nodule between benign or malignant [19].

Early testing with this module presented the following results: the detection scheme of this module yielded 74% of sensitivity with 3.5 false positives per image; the segmentation scheme matches 65.8% against a specialist response and the classification process reached a sensitivity of 81.28% with 20.28% false positive rate. More tests are needed before a full comparison can be drawn to other similar systems.

B.3Calcifications

This module uses a series of convolution filters derived from Chan [20] and Schiabel [21] to detect, in the given image, the location of clustered microcalcifications. The filter is designed to match the format of a calcification, increasing the signal when a match occurs and removing most of the background. This filter is a reworking of Chan's filter for images of variable size.

This system has been shown to have good results with different types of images specially using FFDM [22]. On average, the system has 89% of sensitivity with 6.9 false-positives per image. When evaluating only FFDM images, the false positive rates are reduced to 1.4 per image. As a comparison, the commercially available ImageChecker [1] has 91% of sensitivity with 1.5 false-positives per image and it is restricted to images produced by its own digitizer system.

B.4 Pectoral muscle

This module takes the segmented pectoral muscle from step A.1 and searches for nodules [16]. In the current design, nodules are only detected, since, in most cases, they will be linfonodes. The demarcation of them facilitates the evaluation of spreading cancer by the radiologist.

This system simply finds the regional maximum intensity of the previously segmented pectoral muscle. These points are usually linfonodes.

 ISBN: 978-1-61208-290-5

The results of this module have shown 74.9% of sensitivity and 3.5 false positives per image.

B.5 BI-RADS classification

The system is designed to present a BI-RADS© classification based on all the information obtained in the previous modules. The final purpose is that the results shown by this CADx system constitute a pre-report, so that a radiologist can confirm or change as fit.

C. Comparison to database

Another step in the future development of this CADx system includes integration with BancoWeb [12] database, allowing the radiologist to see example of similar results to those found in the image evaluated.

III. RESULTS

Currently, the CADx scheme is almost fully developed, but has not yet been integrated into a single system. For example, the microcalcification module is fully functional, but it was developed in MATLAB© and has not been converted to JAVA© yet.

Hence, all results shown in this work are direct tests with the different modules, not considering the pre-processing step in the complete system.

The last two modules in the section above are not yet completely developed (BI-RADS classification and Comparison to database), since these two modules require the rest of the program to be fully integrated for development.

When all modules are working and tested the last step in the development will be the interface itself, which will be developed in association with experienced radiologists to add as much useful functionality as possible.

Once the program is complete the validation will occur in two formats. The first will be a comparison to the DDSM database [23] that has over 2600 cases with complete medical and pathological reports. This will be a direct comparison between the program report versus the official medical and pathological report.

The second form will be a comparison between the program and trained radiologists with at least 10 years' experience. This set of tests will compare the system results to a trained radiologist and also the changes in results when the radiologist has access to the results of the CADx system before making his final call.

If the CADx improves the statistical results of the radiologists, it can be considered useful as a second opinion.

IV. CONCLUSION AND FUTURE WORK

The main purpose of this paper was to present a CADx system that has been shown promising results as a prototype. Once the interface is ready and all tools of the system can work automatically, this will be one of the few CADx systems available for general use.

Each tool has been showing promising results, at least equivalent to others reported in the literature. For the future, it is intended to be established in three formats: downloadable software, an internet system integrated to the BancoWeb database and a library in JAVA with all the tools used to develop the system.

ACKNOWLEDGMENT

To FAPESP for the financial support of this project and to Breast Research Group, INESC Porto, Portugal for the INBreast database images.

REFERENCES

1. Hologic. Hologic - Imagechecker Analog Cad. Imagechecker Analog CAD, 2012. Availible in: <Http://Www.Hologic.Com/En/Breast-Screening/Imagechecker/Screen-Film-Cad-Systems/>. [Retrived: 18 Mar 2013].

2. Kodak. Kodak Mammography Cad Engine, 2004. Available in: <Http://Www.Fda.Gov/Medicaldevices/Productsandmedicalprocedures/Deviceapprovalsandclearances/Recently-Approveddevices/Ucm079437.Htm>. [Retrieved: 5 Sep 2013].

3. Schiabel, H., Vieira, M. A. C., and Ventura, L. Preprocessing For Improving Cad Scheme Performance For Microcalcifications Detection Based On Mammography Imaging Quality Parameters. SPIE MI 2009 Diagnosis. Orlando, FL (Usa): . 2009. pp. 72602g-1 - 72602g-12.

4. Romualdo, L. S., Vieira, M. A. C., and Schiabel, H. Enhancement Of Breast Images By Noise Reduction And Mtf Compensation To Improve Microcalcifications Detection. World Congress On Medical Physics And Biomedical Engineering (IFMBE Proceedings, vol. 25iv). Munich (Germany): . 2009. pp. 801-804.

5. Goes, R. F. and Schiabel, H. Computational Adjust Technique To Digital Mammographic Images Based On Digitizer Characteristic Curve. Journal Of Electronic Imaging, vol. 17, no. 4, pp. 043012-1 - 043012-9, 2008.

6. Goes, C. E., Schiabel, H., and Nunes, F. L. S. Evaluation Of Microcalcifications Segmentation Techniques For Dense Breast Digitized Images. Journal Of Digital Imaging, vol. 15, pp. 231-233, 2002.

 ISBN: 978-1-61208-290-5

7. Silva Jr., E. C., Schiabel, H., and Ventura, L. Detection Of Clusters Of Microcalcification Based On Associated Differential And Morphological Filters In Full Mammogram. SPIE MI 2009: Image Processing. FL (USA): . 2009. pp. 72594-72594.

8. Schiabel, H., Santos, V. T., and Angelo, M. F. Segmentation Technique For Detecting Suspect Masses In Dense Breast Digitized Images As A Tool For Mammography Cad Schemes. 23rd Annual Acm Symposium On Applied Computing (Special Track: Computerapplications In Health Care). Fortaleza, CE, Brasil: Acm New York. 2008. pp. 1333-1337.

9. Ribeiro, P. B., Schiabel, H., and Patrocínio, A. C. Improvement In Artificial Neural Networks Performance By The Selection Of The Best Texture Features From Breast Masses In Mammography Images. World Congress On Medical Physics And Biomedical Engineering - IFMBE Proceedings. Seul (South Korea): . 2006. pp. 2321-2324.

10. Patrocínio, A. C., Schiabel, H., and Romero, R. A. F. Evaluation Of Bayesian Network To Classify Clustered Microcalcification. SPIE MI 2004. San Diego: . 2004. pp. 1026-1033.

11. Patrocínio, A. C. et al. Tumoral Mass Classification By Specialists And The Cad Scheme. World Congress On Medical Physics And Biomedical Engineering - Ifmbe Proceedings. Munich (Germany): . 2009. pp. 75-78.

12. Matheus, B. R. N. and Schiabel, H. Online Mammographic Images Database For Development And Comparison Of Cad Schemes. Journal Of Digital Imaging, 2010. vol. 24, no. 4 pp. 500-506, Jun 2011.

13. Gonzales, R. C. and Woods, R. E. Digital Image Processing. New Jersey: Prentice Hall, 2008.

14. Ferrari, R. J., Rangayyan, R. M., Desautels, J. E., and Frère, A. F. Analysis Of Asymmetry In Mammograms Via Diretional Filtering With Gabor Wavelets. IEEE Trans. On Medical Imaging, vol. 20, no. 9, pp. 953-964, Sep 2001.

15. Pun, T. Entropic Thresholding, The New Approach. Computer Graphics And Image Processing, vol. 16, pp. 210-239, Jul 1981.

16. Schiabel, H. and Menechelli, R. C. Automated Characterization Of Secondary Signals Of Breast Cancer To Compose A Module From A Cadx Scheme. 27th International Congress On Computer Assisted Radiology And Surgery (CARS 2013) In 15th International Workshop On Computer-Aided Diagnosis. Heidelberg, Germany: . 2013. pp. 404-404.

17. American College Of Radiology. American College Of Radiology (ACR) Breast Imaging Reporting And Data System Atlas (Bi-Rads® Atlas). American College Of Radiology Website, 2003. Available in: <Http://Www.Acr.Org/Quality-Safety/Resources/Birads/Mammography>. [Retrieved: Aug 2013].

18. Wu, J., Besnehard, Q. and Marchessoux, C. Automatic Classification For Mammogram Backgrounds Based On Bi-Rads Complaxity Definition And On A Multi Content Analisys Framework. Medical Imaging 2011: Image Processing, Proc. of SPIE 2011, Orlando, vol. 7962, pp. 79623f-79623f-12, Fev 2011.

19. Ribeiro, P. B., Romero, R. A. F., Oliveira, P.R, Schiabel, H., and Vercosa, L. B. Automatic Segmentation Of Breast Masses Using Enhanced Ica Mixture Model. Neurocomputing, vol. 120, pp. 61-71, March 2013. ISSN 0925-2312.

20. Chan, H. P., Doi, K., Vyborny, C. J., Lam, K. L. , and Schmidt, R. A. Computer-Aided Detection Of Microcalcifications In Mammograms: Methodology And Preliminary Clinical Study. Investigative Radiology, vol. 23, no. 9, pp. 664-671, 1998.

21. Nunes, F., Schiabel, H., and Goes, C.E. A Computerized Scheme For Detection Of Clusters Of Microcalcifications By Mammograms Image Processing. Medical Biological Engineering Computing, vol. 35, no. 2, pp. 705-705, 1998. ISSN 0140-0118.

22. Matheus, B., Neto, J., and Schiabel, H. Clustered Microcalcification Detection Scheme For Mammographic Images. The 17th International Conference On Image Processing, Computer Vision, & Pattern Recognition. Las Vegas. IPCV pp. 904-908. 2013.

23. Cheng, H.D., Cai, X, Chen, X., Hu, L., and Lou, X. Computer-Aided Detection And Classification Of Microcalcification In Mammograms: A Survey. Pattern Recognition, vol. 36, no. 12, pp. 2967-2971, 2003.

 ISBN: 978-1-61208-290-5

Rice-Planted Area Extraction by RADARSAT Data Using Learning Vector Quantization Algorithm

Sigeru Omatu
Department of Electronics, Information, and Communication Engineering
Osaka Institute of Technology
Osaka, Japan
e-mail:omatu@rsh.oit.ac.jp

***Abstract*—The classification technique using the neural networks has been recently developed. We apply a neural network of Learning Vector Quantization (LVQ) to classify remote sensing data including microwave and optical sensors for estimation of a rice field. The method has capability of a nonlinear discrimination function which is determined by learning. The satellite data were observed before and after planting rice in 1999. Three RADARSAT and one SPOT/HRV data are used in Higashi-Hiroshima City, Japan. RADARSAT image has only one band data, which is difficult to extract a rice field. However, SAR back-scattering intensity in a rice field decreases from April to May and increases from May to June. Thus, three RADARSAT images from April to June are used for this study. The LVQ classification was applied to RADARSAT and SPOT data in order to evaluate rice field estimation. The results show that the true production rate of rice field estimation for RADASAT data by using LVQ was approximately 60% compared with SPOT data. It is shown that the present method is much better compared with SAR image classification by the maximum likelihood (MLH) method.**

***Keywords*-Remote sensing; Synthetic aperture radar; Neural networks; Learning vector quantization; Maximum likelihood method.**

I. INTRODUCTION

Rice is the most important agricultural product in Japan and widely planted in wide places in Japan. A lot of man-power is still necessary to estimate rice field areas every year. Therefore, the development of a system to monitor the rice crop will be welcome. Satellite remote sensing images, such as LAND SATellite Thematic Mapper(LANDSAT TM), or Satellites Pour l'Observation de la Terre Visible High-Resolution data(SPOT HRV), has been expected to be used for estimating a rice field. However, these optical sensors hardly have been able to get necessary data at a suitable time since it may be often cloudy or rainy during the rice planting season in Japan.

On the other hand, space borne synthetic aperture adar penetrates through clouds. Thus, SAR observes a land surface under any weather condition. The back-scattering intensities of C-band SAR images, such as RADAR SATellite(RADARSAT), or European Remote-Sensing Satellite 1(ERS1)/SAR, change greatly from a non-cultivated bare soil condition before rice planting to an inundated condition just after rice planting [1]. In addition, RADARSAT images are rather sensitive to a change of rice biomass in a growing period of rice [2], [3]. Thus, rice area estimation is expected to be realized in an early stage. In previous works, the authors attempted to estimate a rice field using RADARSAT fine-mode data in the same stage [4]. The estimation accuracy of a rice field was approximately 40% by comparing with the estimated area by SPOT multi-spectral data.

In this study, we attempt to detect a rice field from RADARSAT data using Learning Vector Quantization (LVQ) and to compare with accuracy by Maximum LikelyHood (MLH) method . First, we will explain the LVQ algorithm and then we will show the test site and remote sensing data used here. After that, classification methods will be explained and experimental results and discussion. Finally, we will present the conclusion.

II. LEARNING VECTOR QUANTIZATION ALGORITHM

Vector quantization is to represent a data distribution using a set of units, which are called codebook vectors such that a distortion measure is minimized. The LVQ algorithm was proposed by Kohonen [5] in 1997 to find representative vectors among many vectors by learning. In LVQ, only the closest winning unit (using an Euclidean distance) to the current input data is moved toward it at each iteration.

We will show the principle of the LVQ in more detail in what follows. It consists of two layers which are an input layer and a competitive layer as shown in Fig. 1. In the input layer, input data with a dimension n are given. Let us denote the input vector by $\mathbf{X}$ and neurons in the competitive layer are connected to the input vector with weights w_{ji}, $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, M$ where connection weight vector is denoted by $\mathbf{W_j} = (w_{j1}, w_{j2}, \ldots, w_{jn}), j = 1, 2, \ldots, M$ and M is the number of neurons in the hidden layer. Furthermore, we denote the number of cluster by m, the iteration number by t, and total number of iteration by T.

In order to measure a distance between an input vector $\mathbf{X}$ and a weight vector $\mathbf{W_j}$, we adopt a Euclidean norm given by

$$d_j = ||\mathbf{X} - \mathbf{W_j}|| = \sqrt{\sum_{i=1}^{n}(x_i - w_{ji})^2}. \quad (1)$$

We will search a neuron in the competitive layer, which attains the minimum distance and call it as the winning neuron denoted by c, that is,

$$d_c = \min_j d_j = ||\mathbf{X} - \mathbf{W_c}||. \quad (2)$$

 ISBN: 978-1-61208-290-5

If the input vector $\mathbf{X}$ and the winning unit c belong to the same cluster, then the weight vector $\mathbf{W_c}$ will be moved such that it becomes nearer to $\mathbf{X}$, as shown in Fig. 2.

Conversely, if they do not belong the same cluster, the weight vector $\mathbf{W_c}$ will be moved such that it becomes farther from $\mathbf{X}$, as shown in Fig. 3. Therefore, at an iteration t, if the input vector $\mathbf{X}$ and cluster of c belong to the same cluster, then at the next iteration t+1

$$\mathbf{W_j(t+1)} = \mathbf{W_j(t)} + \alpha(t)(\mathbf{X} - \mathbf{W_j(t)}), \quad j = c \quad (3)$$
$$\mathbf{W_j(t+1)} = \mathbf{W_j(t)}, \quad j \neq c. \quad (4)$$

On the other hand, at an iteration t, if the input vector $\mathbf{X}$ and the winning unit c belong to different cluster, then at the next iteration t+1

$$\mathbf{W_j(t+1)} = \mathbf{W_j(t)} - \alpha(t)(\mathbf{X} - \mathbf{W_j(t)}), \quad j = c \quad (5)$$
$$\mathbf{W_j(t+1)} = \mathbf{W_j(t)}, \quad j \neq c. \quad (6)$$

Initial values of weights w_{ji} are determined by using random numbers. The function $\alpha(t)$ means the learning rate and it is set as follows:

$$\alpha(t) = \alpha_0(1 - \frac{t}{T}) \quad (7)$$

where α_0 is a positive initial coefficient of $\alpha(t)$.

III. Test Site and Remote Sensing Data

The test area has a size of about 9.4 by 7.5 km in Higashi-Hiroshima City, Japan, centered at latitude N 34.42, longitude E 132.70. This site is located at the eastern part of Hiroshima City. Three multi-temporal RADARSAT fine-mode (F1F) images, taken on April 8, May 26, and June 19, in 1999 were used as test data. SPOT/HRV multi-spectral data taken on June 21, 1999 were used to generate a reference image for a rice field extraction. Above, three merged RADARSAT, as shown in Fig. 4; one SPOT image in a part of the test site is shown in Fig 5. The rice fields are mainly distributed in the bottom-center portion in the images. The land surface condition in the rice fields on April 8 is a non-cultivated bare soil before rice planting with rather rough soil surface. The surface condition on May 26 is an almost smooth water surface just after rice planting, and that on June 19 is a mixed condition of growing rice and water surface.

The RADARSAT raw data were processed using Vexcel SAR Processor (VSARP) and single-look power images with 6.25 meters ground resolution were generated. Then the images were filtered using median filter with 7 by 7 moving window. All RADARSAT and SPOT images were overlaid onto the topographic map with 1:25,000 scale. As RADARSAT images are much distorted by foreshortening due to topography, the Digital Elevation Model (DEM) with 50 meters spatial resolution issued by Geographical Survey Institute (GSI) of Japan was used to correct foreshortening of RADARSAT images.

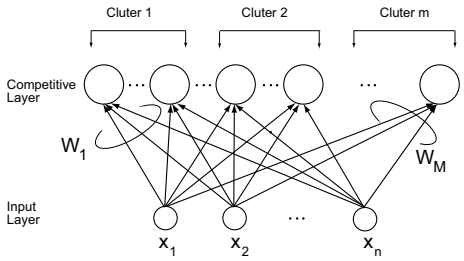


Fig. 1. LVQ structure

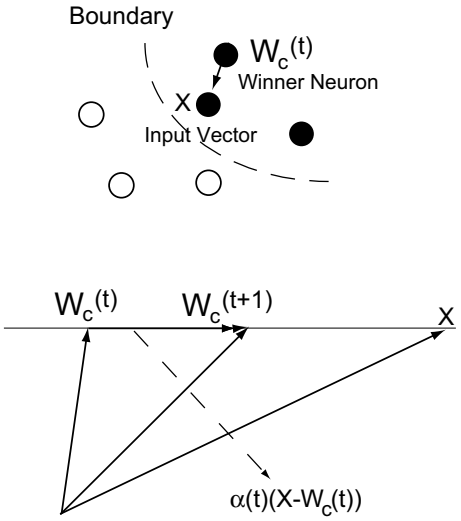


Fig. 2. LVQ learning of weights when the winning neuron belongs to the same cluster of the input $\mathbf{X}$ where $\alpha(t) > 0$.

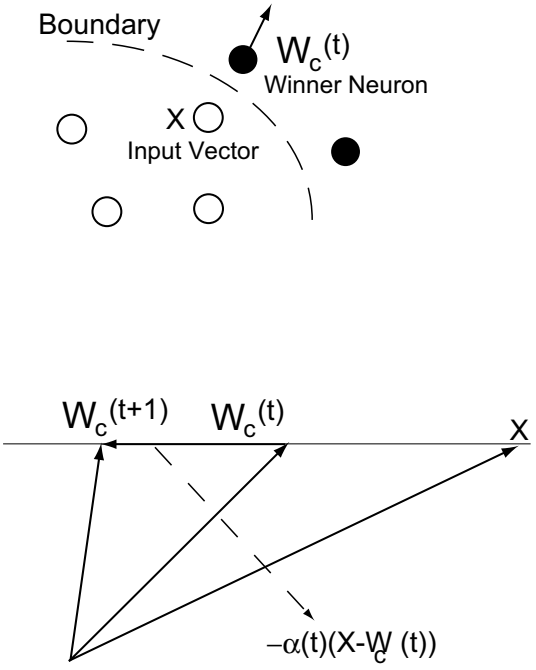


Fig. 3. LVQ learning of weights when the winning neuron belongs to the different cluster of the input $\mathbf{X}$ where $\alpha(t) > 0$.

IV. Classification Methods

Rice fields were extracted using two supervised classification methods from three temporal RADARSAT images and one SPOT image. One was an MLH classifier and the other was an LVQ classifier. In case of the LVQ classifier we adopted the initial weight $\alpha_0 = 0.05$ for RADARSAT and $\alpha_0 = 0.1$ for SPOT. The MLH classifier has been used as a land cover classification for satellite images. However, the classification results may become poor accuracy since it assumes that the distribution of each categorical data is normal distribution. Kohonen's LVQ is a classification method based on competitive neural networks, which allows us to define a group of categories on the space of input data by supervised learning algorithm.

A water region, an urban area, a rice field, and two kinds of forest were selected as target categories. We considered three sub-classes in a water region and an urban area. Furthermore, four sub-classes in a rice field and a forest. Then, we consider them as the same class. The training data for supervised classification was selected by the map and the ground truth.

As training data of SPOT image we selected as ten areas of 5x5 pixels , namely 250 pixels in each category. For the purpose of extraction of a rice field, the training data of the rice field were added 800 pixels to the data.

V. Experimental Results and Discussion

In the beginning, SPOT image as an optical sensor classifies were used for two methods of LVQ and MLH and the classification results were compared. As we can see in Tables I and II, the results of the confusion matrix were examined by SPOT data. Comparing the results of two methods, LVQ was a little better at high accuracy level, although the differences were not so large.

The results of classification rate were shown in Tables III and IV. The classification score of a rice field by SPOT was about 90 % for both of two classification methods and RADARSAT was about 80 % accuracy. Table IV shows the result by LVQ classifier of multi-temporal RADARSAT data. The results are better than MLH. In particular, the rice classification accuracy of LVQ is much better than MLH. Tables IV shows the results of average accuracy. The LVQ classifier is superior to the MLH clasifier.

Figures 6 and 7 show the classified images by RADARSAT and SPOT, respectively. The rice fields were obtained by the three temporal RADARSAT images. The classification result of the urban and the forest area were different between these two images, on the other hand, water and rice areas were resembled between these two images.

We defined two indices, a True Production Rate (TPR) and a False Production Rate (FPR) for rice areas by RADARSAT. TPR is the coincidence rate of rice areas by RADARSAT within those by SPOT and FPR is the rate of non-rice areas by SPOT within rice areas by RADARSAT. As the rice area images extracted by RADARSAT are still contaminated by speckle noises, the majority filter with 7 by 7 window was applied to the rice extracted images by RADARSAT before evaluation. The rice extracted image by SPOT was also filtered by the same majority operation as RADARSAT to make the ground resolution compatible each other, as shown in Table V.

We found experimentally that about 60 % of rice areas by SPOT were not extracted by RADARSAT and about 35 % of the areas by RADARSAT were outside areas of rice by SPOT using LVQ. This result of TPR was better than that of MLH. Figure 6 shows the results of extracted rice field in part of the test site. In the figure, the white region shows the rice field of each image. MLH has not included adjustable parameters by users compared with LVQ. The latter could more excellent results we must fine tuning parameters, especially the learning rate $\alpha(t)$ selection needs trial error to get the better results.

Figure 8 and Figure 9 show the results of extracted rice field in part of test site of RADASAT by MLH and by the LVQ method where white color denotes rice-planted area. Figure 10 show the results of extracted rice field in part of test site of SPOT by LVQ. From these results, the classification result by LVQ becomes more detailed extraction of rice fields compared with MLH.

VI. Conclusions

A rice field extraction was attempted using multi-temporal RADARSAT data taken in the early stage of rice growing season by MLH and LVQ classifications. The LVQ classification is much better compared with classification by the MLH for a rice field extraction by RADARSAT data. However, for a quantitative evaluation, the rice field areas by RADARSAT resulted in poor coincidence rate with those by SPOT. Thus, we will apply this proposed method to other SAR data due to the extraction rice field.

Acknowledgment

This work was supported by JSPS KAKENHI Grant-in-Aid for Scientific Research (B) (24360141). The authors would like to thank JSPS to support this research work.

References

[1] Y. Suga, Y. Oguro, and S. Takeuchi, "Comparison of Various SAR Data for Vegetation Analysis over Hiroshima City", Advanced Space Research , vol. 23, no. 8, August, 1999, pp. 225–230.

[2] M. Bicego and T. L. Toan, "Rice Field Mapping and Monitoring with RADARSAT Data", International Journal of Remote Sensing, vol. 20, no. 4, April, 1999, pp. 745–765.

[3] S. C. Liew, and P. Chen, "Monitoring Changes in Rice Cropping System Using Space-borne SAR Imagery", Proceedings of IGARSA'99, October, 1999, pp. 741–743.

[4] Y. Suga, S. Takeuchi, and Y. Oguro, "Monitoring of Rice-Planted Areas Using Space-borne SAR Data", Proceedings of IAPRS, XXXIII, B7, February, 2000, pp. 741–743.

[5] T. Kohohnen, "Self-Organizing Maps", Springer, 1997, pp. 206–217.

 ISBN: 978-1-61208-290-5

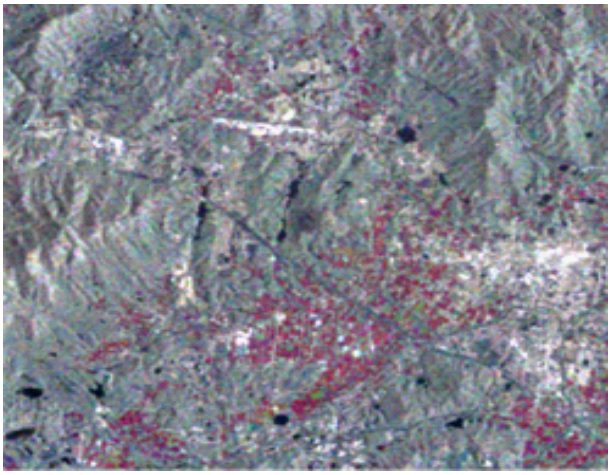


Fig. 4. RADARSAT F1F mode image in the test site.CSA & RADARSAT International 1999.

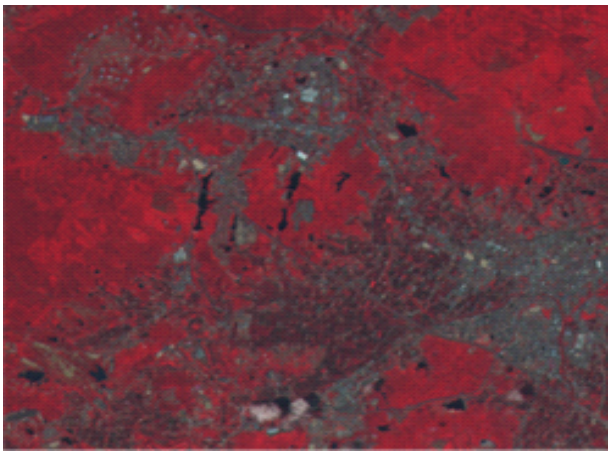


Fig. 5. SPOT-2HRV image in the test site. CNESS 1999.

TABLE I
THE CONFUSION MATRIX FOR THE CLASSIFICATION USING THE MLH(SPOT)(%)

	Water	Urban	Rice	Forest
Water	100.0	0.0	0.0	0.0
Urban	0.0	100.0	0.0	0.0
Rice	0.0	0.0	100.0	0.0
Forest	0.0	0.0	0.0	100.0

TABLE II
THE CONFUSION MATRIX FOR THE CLASSIFICATION USING THE LVQ(SPOT)(%)

	Water	Urban	Rice	Forest
Water	100.0	0.0	0.0	0.0
Urban	0.0	100.0	0.0	0.0
Rice	0.0	0.0	100.0	0.0
Forest	0.0	0.0	0.0	100.0

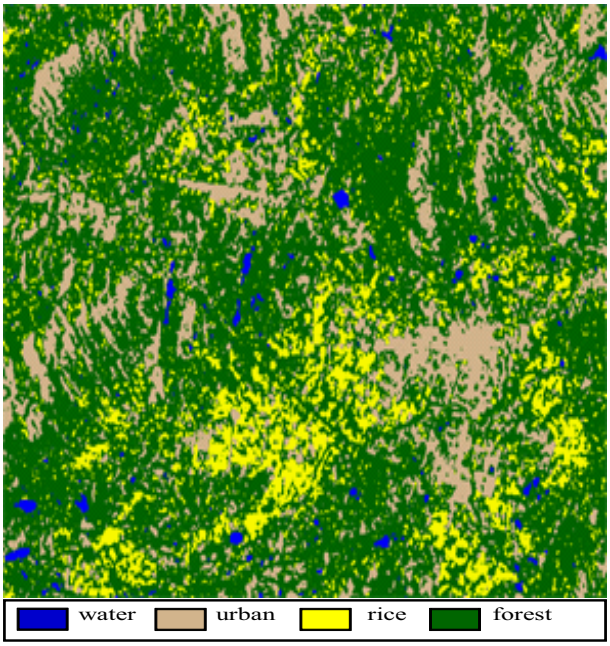


Fig. 6. Classification result of RADARSAT F1F image by LVQ.

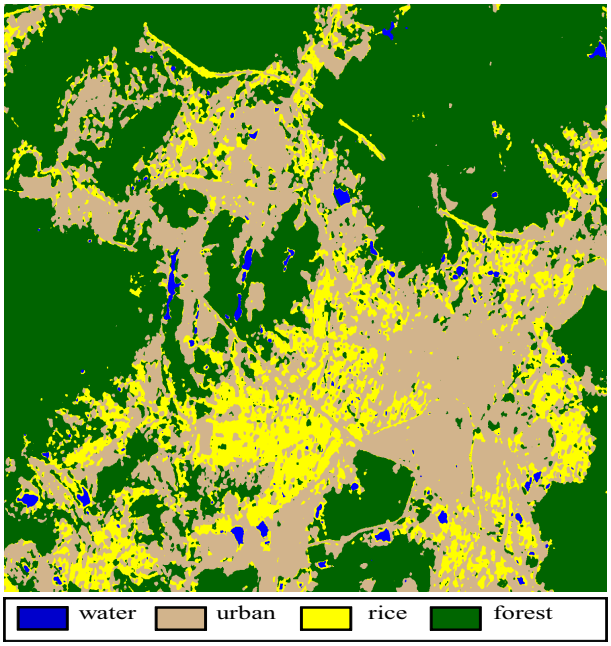


Fig. 7. Classification result of SPOT/HRV image by LVQ.

 ISBN: 978-1-61208-290-5

TABLE III
THE CONFUSION MATRIX FOR THE CLASSIFICATION USING THE MLH (RADARSAT) (%)

	Water	Urban	Rice	Forest
Water	100.0	0.0	0.0	0.0
Urban	0.0	62.0	0.0	38.0
Rice	0.0	4.5	53.3	42.2
Forest	2.0	3.6	9.2	85.2

TABLE IV
THE CONFUSION MATRIX FOR THE CLASSIFICATION USING THE LVQ (RADARSAT) (%)

	Water	Urban	Rice	Forest
Water	100.0	0.0	0.0	0.0
Urban	0.0	71.2	0.0	28.8
Rice	0.0	0.0	87.2	12.8
Forest	2.8	6.6	2.4	88.2

TABLE V
RESULT OF RICE FIELD EVALUATION BY RADARSAT COMPARED WITH SPOT BY LVQ. (%)

Method	TPR(%)	FPR(%)	TPR-FPR(%)
MLH	46.7	24.6	22.1
LVQ	59.1	62.0	35.6

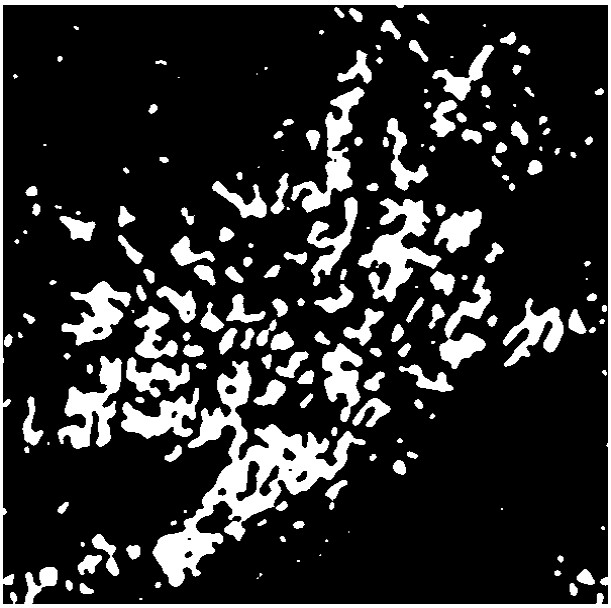


Fig. 8. Results of extracted rice field in part of test site of RADASAT by MLH.

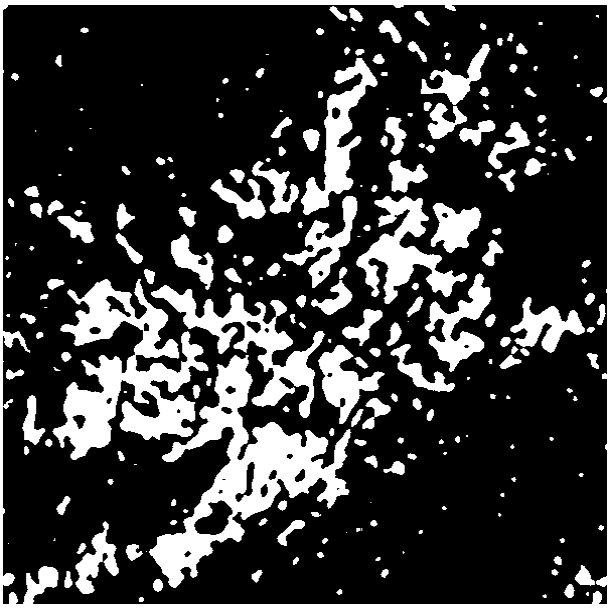


Fig. 9. Results of extracted rice field in part of test site of RADASAT by LVQ.

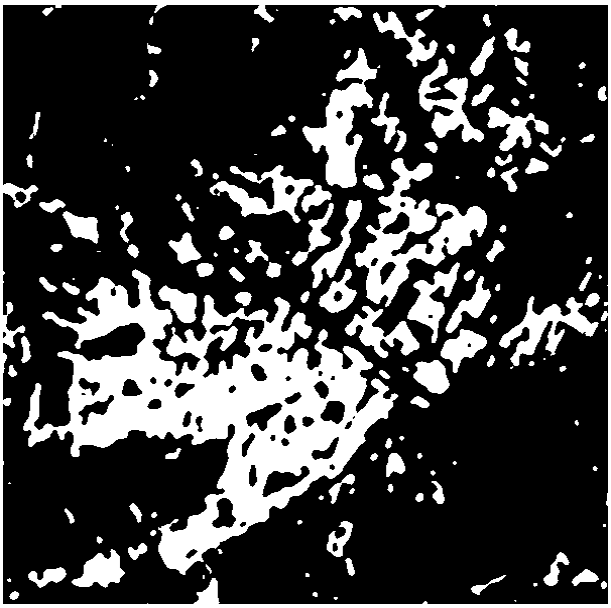


Fig. 10. Results of extracted rice field in part of test site of SPOT by LVQ.

 ISBN: 978-1-61208-290-5

A UsiXML Proposal for a Pattern-Oriented and Model-Driven Architecture for Interactive Systems

Mohamed Taleb
Department of Software Engineering & Information Technology
Software Engineering Research Laboratory
École de technologie supérieure, University of Quebec, Montreal, Quebec, Canada
mohamed.taleb.1@ens.etsmtl.ca

Ahmed Seffah
Depatment of Computer Science and Software Engineering
Human-Centered Software Engineering Group
Concordia University, Montreal, Quebec, Canada
seffah.ahmed@yahoo.fr

Alain Abran
Department of Software Engineering & Information Technology
Software Engineering Research Laboratory
École de technologie supérieure, University of Quebec, Montreal, Quebec, Canada
alain.abran@etsmtl.ca

***Abstract*—Despite its obvious and well-publicized potential to support the model-driven engineering of user interfaces, the (re)use of the rich variety of Human-Computer Interaction (HCI) design patterns, we have today has not achieved the acceptance and widespread applicability of HCI design patterns within the existing model-driven engineering framework. This paper proposes a specification and a User Interface eXtensible Markup Language (UsiXML)-based formalization of a unifying Pattern-Oriented and Model-driven Architecture (POMA). We have already introduced a set of extensions, called the POMA Markup Language (POMAML), designed to facilitate the specification of all the intrinsic components of the POMA architecture, including its patterns and models, and the relationships between these two artifacts within the model.**

Keywords-Pattern-Oriented; Model-Driven Architecture; UsiXML; User Interface; POMA.

I. INTRODUCTION

Day-to-day experience suggests that it is not enough to approach a complex design with a set of models and model-driven engineering languages and tools. The developers must also be able to use (reuse) proven solutions emerging from the best model-driven practices for building models and their transformations, as well as for generating code for diverse platforms.

Without these solutions, developers are unable to properly define valid models, and so cannot take full advantage of the power of the model orientation, resulting in poor performance. Invalid models will lead to poor scalability and usability. Furthermore, the designer might find himself "reinventing the wheel" when attempting to develop an application.

We propose to enhance, extend, or rethink the activities and artifacts of the model-driven engineering frameworks using patterns for model construction, transformation, and mapping. We proposed POMA (Pattern-Oriented and Model-driven Architecture) [1] as a unifying architecture to bridge the gap between patterns and models, as well as between the model-driven engineering and pattern-oriented design frameworks.

Specifically, we consider the possible extensions to the User Interface eXtensible Markup Language (UsiXML) collection of models [2] and the four basic levels of model abstraction defined in the Cameleon Reference Framework [2]. UsiXML defines, validates, and standardizes an open User Interface Description Language, while increasing the productivity and reusability of multi-platform and multi-context interactive applications. It also improves the usability and accessibility of these applications.

In our ongoing research, we are aiming at specifying and representing the components of the POMA architecture. We suggest extensions to the concepts of UsiXML to formalize a language called POMAML (Pattern-Oriented and Model-driven Architecture Markup Language). In other words, POMA is a unifying architecture to bridge the gap between patterns and models using POMAML.

This paper is organized as follows. Section 2 introduces related work on POMA fundamentals, the basic concepts of the POMA architecture, and the basic structural notation of UsiXML. Section 3 primarily describes the application of UsiXML in the POMA architecture. Section 4 presents an illustrative case study. Section 5 presents a summary and directions for future work.

II. RELATED WORK

Over the past two decades, research on interactive system and User Interface (UI) engineering has resulted in several architectural models, which constitute a major contribution not only to facilitate the development and maintenance of interactive systems, but also to promote the standardization, portability, and ergonomic usability (ease of use) of the interactive systems developed. Such architectures provide a clear separation of concerns [3]. In particular, they decouple the UI from the system semantics, and define the reusable and the standardized UI components.

A number of UI languages and notations have been suggested to specify architecture and model user interfaces for different platforms and at different levels of abstraction. For example, User Interface Markup Language (UIML) [4] is a meta-language that allows the developer to describe the UI in generic terms and to use style descriptions to map the UI to various target platforms. UIML was developed to address the need for a uniform UI description language for building multi-platform systems. eXtensible User-interface Language (XUL) [5; 6] is an official Mozilla initiative,

 ISBN: 978-1-61208-290-5

which provides an XML-based language for describing window layout. The goal of XUL is to build cross platform systems that are easily portable to all the operating systems on which Mozilla runs. XUL provides a clear separation between the UI definition (the various widgets that make up the UI) and its visual appearance (the layout and the "look and feel").

EXtensible Interface Markup Language (XIML) followed a declarative interface modeling language called MIMIC [7], and provides a way to describe the UI without worrying about its implementation. The aim of XIML is to describe the various abstract aspects (domain, task, and user) and concrete aspects (presentation and dialog) of the UI throughout the development life cycle. In addition, XIML supports the definition of mapping from abstract elements to concrete elements [8].

TERESA [9] provides tools to allow developers to interactively define mappings between the various models. Web Services eXtensible Markup Language (WSXML) [10] integrates Web services and XML into the Service Oriented Architecture. Web services constitute a technological approach that is well suited to bridge information systems, and can enable this integration, even when systems are implemented on disparate platforms or through differing technologies. XML is useful for a variety of data exchange applications, and is a foundation technology for such enterprise strategies as Web services, and likely has a future in enterprises.

GrafiXML, developed by Limbourg et al. [2], is an original UI builder, in that it enables designers and developers to design several UIs simultaneously for multiple contexts of use, i.e., for many users, platforms, and environments. GrafiXML is an intelligent UI builder, in that it maintains model consistency between these representations through a set of mappings based on the UI ontology.

Following the lead of the object-oriented software design community, HCI practitioners investigated design patterns as one possible way to capture and use the best design practices. An HCI design pattern is defined as a named, reusable solution to a recurring user problem in different contexts of use, including the various computing platforms (Web, Graphical User Interface (GUI), mobile applications, etc.). Relationships between patterns have been explored to combine related patterns into pattern languages, resulting in a lingua franca for design [12].

Hundreds of HCI design patterns are freely available on the Web. However, providing a list of patterns and their loosely defined relationships, as is done for most HCI pattern languages, is insufficient for effectively driving design solutions. Understanding when a pattern is applicable during the design process and how it can be used, as well as how and why it can or cannot be combined with other related patterns, are key notions in the application of patterns.

Javahery and Seffah [3] proposed a design approach, called Pattern-Oriented Design (POD), which provides a framework for guiding designers through stepwise design suggestions. At each predefined design step, designers are given a set of applicable patterns. This process is in stark contrast to the current use of pattern languages, where there is no defined link to any sort of systematic method. Pattern relationships are explicitly described, which allows designers to compose patterns based on an understanding of these relationships. In POD, patterns are building blocks at different levels of abstraction, which makes them extremely useful for designers when driving the UI design based on user experiences [14; 15; 3].

The proposed Pattern-Oriented and Model-driven Architecture (POMA) (Figure 1) [1] identifies an extensive list of pattern categories and types of models aimed at providing a pool of proven solutions to these problems. The models of patterns span several levels of abstraction, such as domain, task, dialog, presentation, and layout. The proposed POMA architecture illustrates how several individual models can be combined at different levels of abstraction into heterogeneous structures, which can then be used as building blocks in the development of interactive systems.

The various components of the POMA architecture are detailed in [1], and include:

- The architectural levels and various categories of patterns [16], [17], and [19];
- The Platform Independent Model (PIM) and Platform Specific Model (PSM) [18];
- The pattern composition rules for selecting and composing patterns corresponding to each type of PIM model [16] and [18];
- The rules for mapping patterns and PIM models to produce PSM models for multiple platforms [16] and [18];
- The rules for transforming PIM to PIM models and PSM to PSM models [20];
- The rules for source code generation;
- The generation of the whole of application.

The rationale and strengths of the POMA architecture are as follows:

- POMA facilitates the use of patterns by beginners as well as experts;
- POMA supports the automation of both the pattern-driven and model-driven approaches to design;
- POMA supports the communication and reuse of individual expertise regarding good design practices;
- POMA can integrate all the new technologies, including traditional office desktops, laptops, Palmtops, PDAs (with or without keyboards), mobile telephones, and interactive televisions, among others.

 ISBN: 978-1-61208-290-5

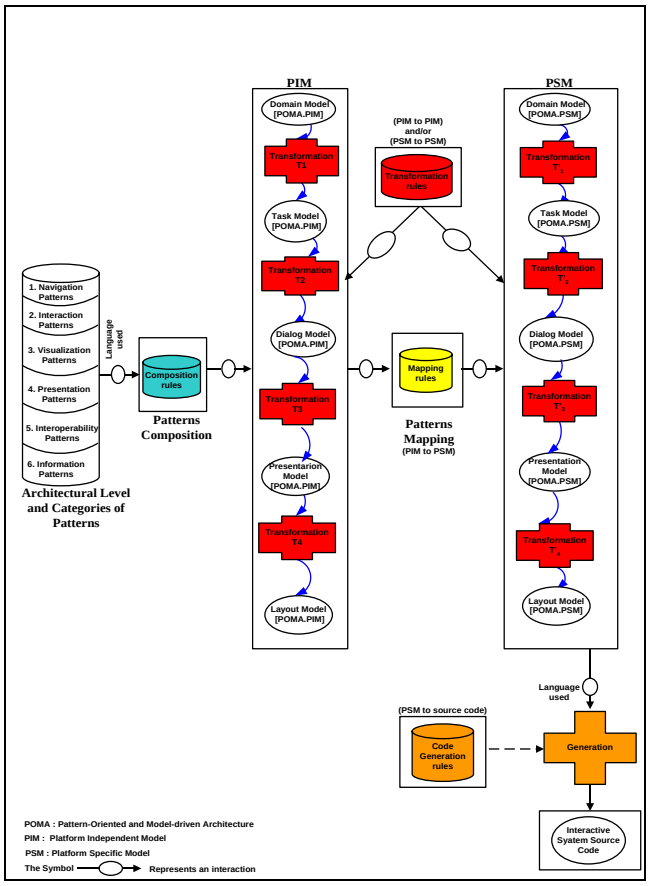


Figure 1. POMA architecture for interactive system development [1].

The User Interface eXtensible Markup Language (UsiXML) [21; 24] is an XML-compliant markup language. It is an approach for describing the structure and presentation aspects of the UI to describe dialog modeling [22]. Limbourg et al. [2] describe the structured UsiXML, based on the following four basic levels of abstraction defined in the Cameleon reference framework. This framework is intended to represent the UI development life cycle for context-sensitive interactive applications. In other words, the framework defines UI development steps for multi-context interactive applications. It structures development processes for two contexts of use into four development steps (each development step being able to manipulate any specific artefact of interest as a model or a UI representation):

1. Task and Concepts (the highest level), where the user task is defined based on his viewpoint, along with the various objects that are manipulated by it.
2. Abstract User Interface (AUI): abstracts the Concrete User Interface (CUI) into a UI definition that is interaction modality independent (e.g., graphical/vocal interaction);
3. Concrete User Interface (CUI): abstracts the Final User Interface (FUI) into a UI definition that is independent of any computing platform;
4. Final User Interface (FUI): a UI running on a particular platform, either by interpretation or by execution.

UsiXML is defined as a set of XML schemas, each corresponding to one of the models within the scope of the language. It consists of a User Interface Description Language (UIDL), which is a declarative language capturing the essence of what a UI is, or should be, independently of physical characteristics. It describes the constituent elements of the UI of an application at a high level of abstraction: widgets, controls, containers, modalities, interaction techniques, etc. Despite that, UsiXML does not require the use of any particular development process, which means that designers are free to choose the most appropriate abstraction level at which to begin their projects [23].

III. POMA COMBINED WITH THE USIXML APPROACH

To tackle some of the weaknesses identified in related work, a set of UsiXML concepts proposes to specify and formalize the POMA architecture within the UsiXML perspective (Figure 3) and its language, which is called the POMA Markup Language (POMAML) and is described in section 4 (Pattern-Oriented Modeling Architecture Markup Language). The formalization is achieved in visual, structural, and formal notations using XML for modeling the patterns and models of the POMA components described in section II in order to generate the specifications for various types of UI engineered for interactive systems. Our aim is to persevere with this objective, and continue to design and reuse POMA architecture specifications that span different levels of abstraction, such as the domain, task, dialog, presentation, and layout models, until the final layout of the various UIs has been generated.

Because of the number of concepts it embodies, UsiXML is used to illustrate the POMA architecture (Figure 3). On the left is a series of development steps that comply with the Cameleon reference framework [22], and on the right are the concepts supported by UsiXML, and the transformations and mappings applied to it. POMA architecture based on UsiXML classifies UIs for supporting a target platform and a context of use, and enables to structure the development life cycle into five levels of abstraction and patterns categories as follows (Figure 2):

1. Categories patterns library. These patterns of different categories are defined and formalized in XML language;
2. Five categories of models in PIM and PSM (Task, Domain, Dialog, Presentation, Layout) used in POMA architecture, providing examples, for a model-driven architecture for interactive systems to resolve many recurring design problems, examples of which include: (1) decoupling the various aspects of Web applications such business logic, the user interface, navigation and information architecture; (2)

 ISBN: 978-1-61208-290-5

isolating platform-specific problems from the concerns common to all interactive systems.

3. Abstract User Interfaces (AUI) of PIM. This abstraction level defines a generic user interface description of PIM models completely independent of the considered UI toolkit and multi-platforms.
4. Concrete User Interface (CUI) Platform Independent Model (PSM) for different platforms (Laptop, PDA, Cellular, Palmtop, interactive television, iPhone, etc.). This level defines the graphical concrete user interfaces, including the concrete interaction objects (CIO), for each specific platform.
5. Final User Interface (FUI). This level is to generate the source code of the entire application for a specific platform.

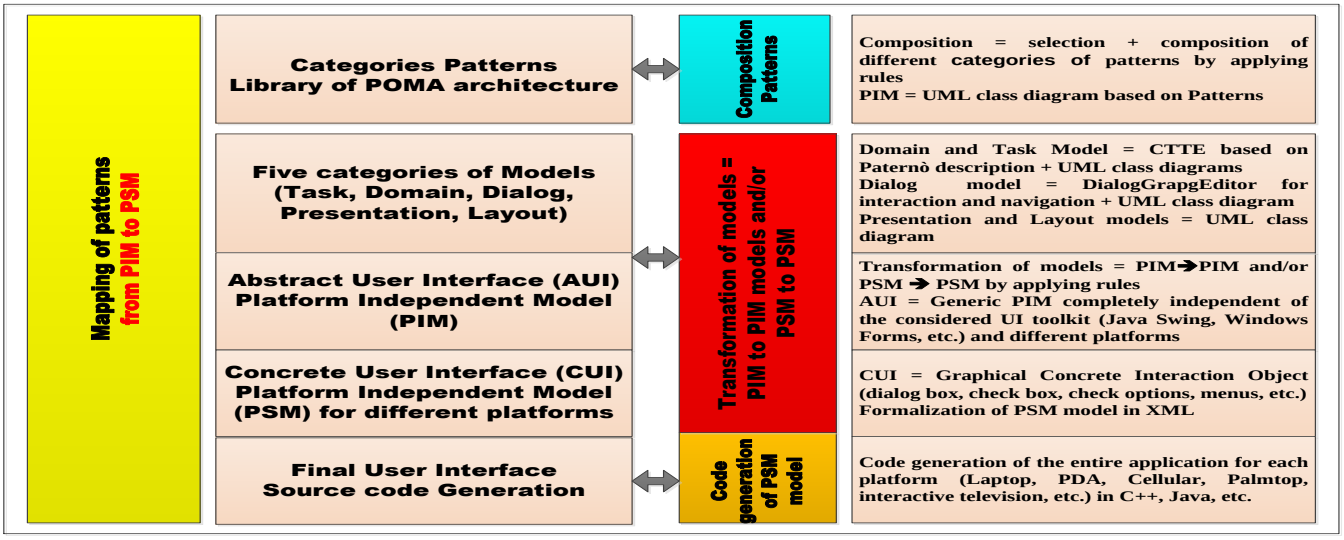


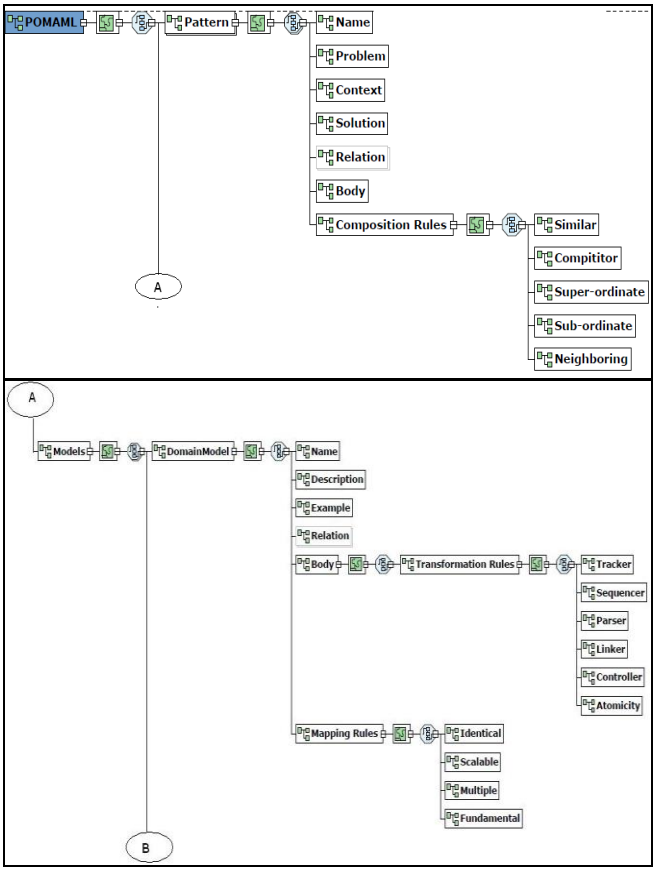
Figure 2. POMA architecture in UsiXML perspective.

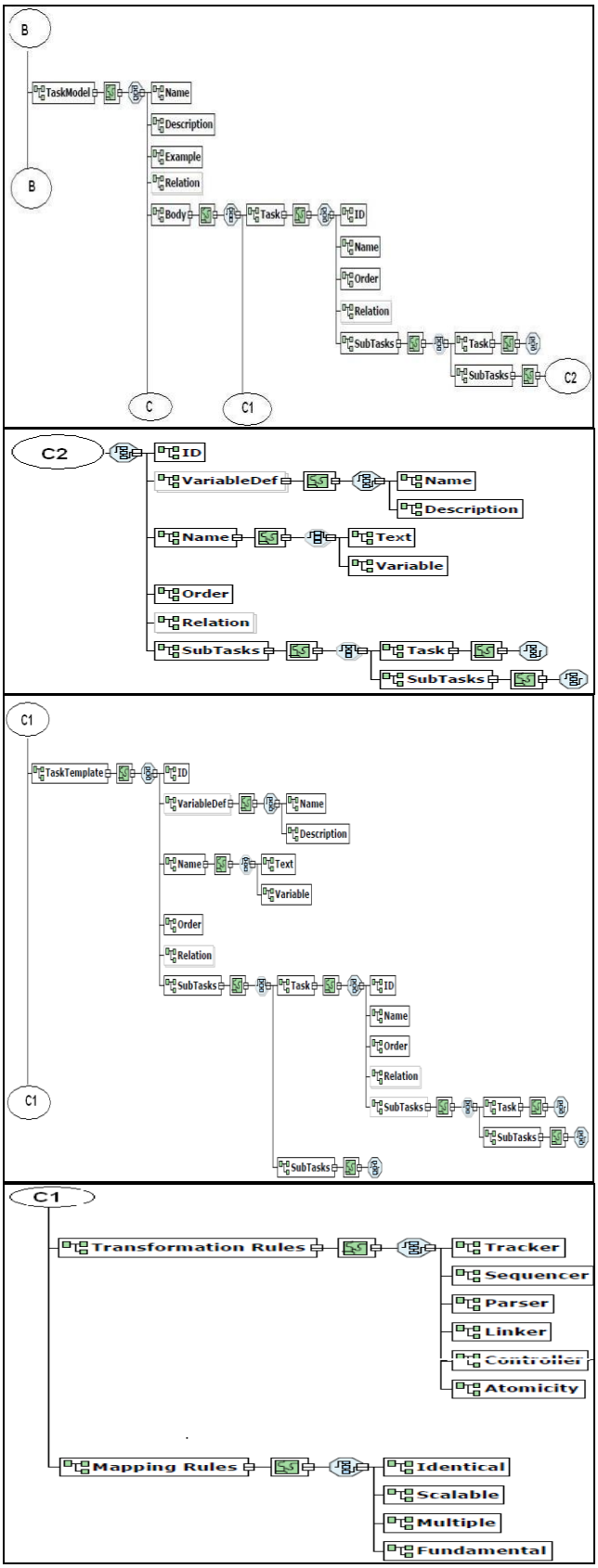
In this section, we describe a design that illustrates and clarifies the core ideas underlying the approach combining the POMA architecture with the UsiXML, and explain its practical relevance. The proposed POMA architecture combined with UsiXML (Figure 2) shows how UsiXML concepts are used to represent the components of the POMA architecture to generate the source code of the various concrete UIs of the application.

With the POMA architecture, it is possible to design a formalism to describe a software architecture based on the composition of several patterns to generate different types of applications. This formalism can take three forms:

- Structural, using the XML formalization language called POMAML;
- Formal, using mathematical methods and concepts;
- Visual, using UML specifications such as sequence diagrams and class diagrams.

Here, we focus essentially on the use of the structural notation to describe the entire POMAML language (Figure 3) of the POMA architecture components, such as patterns, composition rules, levels of PIM and PSM models, transformation rules, mapping rules, and generation rules based on the XML notation.





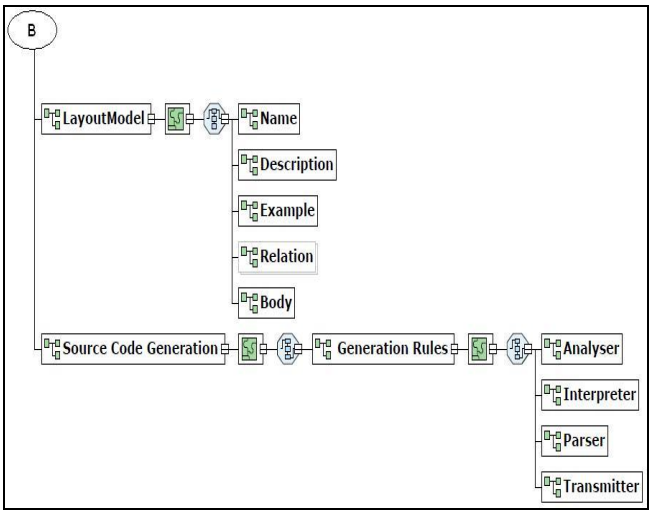


Figure 3. POMAML language

IV. AN ILLUSTRATIVE CASE STUDY

This following example presents the domain model (Figure 4) of the POMA architecture for a laptop platform using UsiXML concepts. In this case, the more those high-level tasks are decomposed, the easier it is to use the reusable task structures that have been obtained or captured from other projects or systems. Here, these reusable task structures are documented in the form of patterns. This approach ensures an even greater degree of reuse.

```
<xs:element name="DomainModel">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Name" type="xs:string"/>
      <xs:element name="Description" type="xs:string"/>
      <xs:element name="Example" type="xs:string"/>
      <xs:element name="Relation" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="Body">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="Transformation Rules">
              <xs:complexType>
                <xs:sequence>
                  <xs:element name="Tracker"/>
                  <xs:element name="Sequencer"/>
                  <xs:element name="Parser"/>
                  <xs:element name="Linker"/>
                  <xs:element name="Controller"/>
                  <xs:element name="Atomicity"/>
                </xs:sequence>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="Mapping Rules">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="Identical"/>
            <xs:element name="Scalable"/>
            <xs:element name="Multiple"/>
            <xs:element name="Fundamental"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Figure 4. Domain Model in POMAML language

V. SUMMARY AND FUTURE WORK

In this paper, we have discussed a perspective from which the POMA architecture is specified and represented using the UsiXML approach. Previously, we had provided a set of extensions, called POMAML, which makes it possible to generate source code in different programming languages for each platform of an interactive system.

Our research has resulted in the integration and formalization of UsiXML for the POMA architecture. It has also led to avenues for further research, such as:

- Description of a process for the generation of source code from POMA's five PSM models;
- Development of a tool that automates the POMA architecture-based engineering process;
- Standardization of the POMA architecture to all types of systems, and not only to multi-platform interactive systems;
- Quality assurance of the applications produced, since a pattern-oriented architecture will also have to provide for the encapsulation of quality attributes and facilitate prediction;
- Validation of the migration, usability, and overall quality of the POMA architecture for interactive systems using existing methods;
- Evaluation of the effectiveness and learning time of the POMA architecture for both novices and expert users.

REFERENCES

[1] M. Taleb, A. Seffah, and A. Abran, "Interactive Systems Engineering: A Pattern-Oriented and Model-Driven Architecture", The 2009 International Conference on Software Engineering Research and Practice (SERP'09), July 2009, pp. 636-642, Las Vegas, USA.

[2] Q. Limbourg, J. Vanderdonckt1, B. Michotte1, L. Bouillon1, and V. López-Jaquero, "USIXML: A Language Supporting Multi-path Development of User Interfaces", vol. 3425/2005, *Engineering Human Computer Interaction and Interactive Systems*, July 2005, pp. 200-220, DOI 10.1007/b136790, ISBN 978-3-540-26097-4, Springer Berlin/Heidelberg Publisher.

[3] H. Javahery and A. Seffah, "A Model for Usability Pattern-Oriented Design", in Proceedings of TAMODIA2002, July 2002, pp. 104-110, Bucharest, Romania.

[4] M. Abrams, C. Phanouriou, A. L. Batongbacal, S. M. Williams, and J. E. Shuster, "UIML: An Appliance-Independent XML User Interface Language", Proceedings of the 8th International WWW Conference, May 1999, pp. 1695–1708, Elsevier Science Publishers, Toronto, Canada.

[5] XUL, The XML User Interface Language, 2004, at: http://www.xulplanet.com, [retrieved: April, 2013].

[6] XUL, XUL Tutorial, 2004, at: http://www.xulplanet.com/tutorials/xultu, [retrieved: April, 2013].

[7] A. Puerta and D. Maulsby, "Management of Interface Design Knowledge with MODI-D", in Proceedings of IUI'97, January 1997, pp. 249-252, Orlando, FL, USA.

[8] A. Puerta and J. Eisenstein, "Towards a General Computational Framework for Model-Based Interface Development Systems", in Proceedings of IUI'99, January 1999, pp. 171-178, Los Angeles, CA, ACM Press, New York, USA.

[9] TERESA, Transformation Environment for Interactive Systems Representation, 2004, at: http://giove.cnuce.cnr.it/teresa.html, [retrieved: April, 2013].

[10] P. Classon and J. Prem, "SOA: Integrating XML and Web Services," LiquidHub Inc., 2004, at: http://www.liquidhub.com/docs/Horizons_WSXML_primer_v3.pdf, [retrieved: April, 2013].

[11] B. Michotte and J. Vanderdonckt, "GrafiXML, A Multi-Target User Interface Builder based on UsiXML", Fourth International Conference on Autonomic and Autonomous Systems, IEEE Computer Society, March 2008, pp. 15-22, DOI 10.1109/ICAS.2008.29, ISBN 0-7695-3093-1/08.

[12] T. Erickson, "Lingua Franca for Design: Sacred Places and Pattern Language", in Proceedings of Designing Interactive Systems, August 2000, pp. 357-368, ACM Press, New York (NY), USA.

[13] J. O. Borchers, "Pattern Approach to Interaction Design", Proceedings of the DIS 2000 International Conference on Designing Interactive Systems, August 2000, pp. 369-378, ACM Press, New York, USA.

[14] A. Granlund and D. Lafrenière, "A Pattern-Supported Approach to the User Interface Design Process", Workshop Report, UPA'99 Usability Professionals' Association Conference, June-July 1999a, Scottsdale, AZ.

[15] M. Taleb, H. Javahery, and A. Seffah, "Pattern-Oriented Design Composition and Mapping for Cross-Platform Web Applications, the 13th International Workshop, DSV-IS 2006, vol. 4323/2007, Springer-Verlag, July 2006, Trinity College, Dublin Ireland, DOI 10.1007/978-3-540-69554-7, ISBN 978-3-540-69553-0, Berlin Heidelberg, Germany.

[16] M. Taleb, A. Seffah, and A. Abran, "Pattern-Oriented Architecture for Web Applications", 3rd International Conference on Web Information Systems and Technologies (WEBIST 2007), March 2007, pp. 117-121, ISBN 978-972-8865-78-8, Barcelona, Spain.

[17] M. Taleb, A. Seffah, and A. Abran, "Model-Driven Design Architecture for Web Applications", The 12th International Conference on Human Centered Interaction International (FIC-HCII 2007), Beijing International Convention Center, Beijing, P. R. China, vol. 4550/2007, July 2007, pp. 1198-1205, Springer-Verlag, Berlin Heidelberg, Germany.

[18] M. Taleb, A. Seffah, and A. Abran, "Pattern-Oriented Design for Cross-Platform Web-based Information Systems", The 2007 IEEE International Conference on Information Reuse and Integration (IEEE IRI-07), August 2007, pp. 122-127, Las Vegas, USA.

[19] M. Taleb, A. Seffah, and A. Abran, "Transformation Rules in POMA architecture", The 2010 International Conference on Software Engineering Research and Practice (SERP'10), July 2010, pp. 636-642, Las Vegas, USA.

[20] UsiXML, What is UsiXML?, Université catholique de Louvain, Belgium, 2007, at: http://www.usixml.org/index.php?mod=pages&id=2, [retrieved: April, 2013].

[21] Q. Limbourg, J. Vanderdonckt, B. Michotte, L. Bouillon, M. Florins, and D. Trevisan: "UsiXML: A User Interface Description Language for Context-Sensitive User Interfaces", in Proceedings of the AVI'2004 Workshop on Developing User Interfaces with XML: Advances on User Interface Description Languages, UIXML'04, Gallipoli, Italy, EDM-Luc, May 2004, pp. 55–62.

[22] M. Winckler, F. M. Trindade, A. Stanciulescu, and J. Vanderdonckt, "Cascading Dialog Modeling with UsiXML," Proceedings of the 15th Int. Workshop on Design, Specification, and Verification of Interactive Systems DSV-IS'2008, Kingston, Canada, Lecture Notes in Computer Sciences, vol. 5136, Springer, Berlin, July 2008, pp. 121-135.

[23] Cover Pages (website hosted by OASIS): online resource for markup language technologies, "User Interface eXtensible Markup Language (UsiXML), 2005, at: http://xml.coverpages.org/userInterfaceXML.html#usixm, [retrieved: April, 2013].

 ISBN: 978-1-61208-290-5

A MapReduce Implementation of the Genetic-Based ANN Classifier for Diagnosing Students with Learning Disabilities

Tung-Kuang Wu[1], Shian-Chang Huang[2],
Hsiu-Ting Kao[3], Hsu Chang[4]
Dept. of Information Management, NCUE
Changhua City, Taiwan
[1]tkwu@im.ncue.edu.tw, [2]shhuang@cc.ncue.edu.tw,
[3]b9456005@gmail.com, [4]zx1986@gmail.com

Ying-Ru Meng
Dept. of Special Education, NHCUE
HsinChu City, Taiwan
myr321@mail.nhcue.edu.tw

***Abstract*—Diagnosis of students with learning disabilities (LD) is a difficult procedure that requires extensive man power and takes a long time. Fortunately, through genetic-based (GA) parameters optimization, artificial neural network (ANN) classifier may be a good alternative to the above procedure. However, GA-based ANN model construction is computation-intensive and may take quite a while to process. Accordingly, parallel processing such as multi-core programming and grid computing have been used to speedup the process. In this study, we setup a Hadoop min-cloud environment with virtualized hosts so that we may take full advantage of the current multi-core CPU technology. The GA-based ANN LD classifier is then re-programmed based on the MapReduce programming model and ported to this mini-cloud environment. Some implementation issues and considerations regarding the process will be discussed in the paper. Although the preliminary results may not show significant breakthrough over our previous studies, yet we do gain some experience through this process and see the potential of the MapReduce model in our future applications.**

Keywords-learning disabilities; MapReduce; neural network; virtualization; cloud computing

I. INTRODUCTION

The term "learning disabilities" (LD) was first used in 1963 [1]. However, experts in this field have not yet completely reach an agreement on the definition of LDs and its exact meaning [2]. In fact, a person can be of average or above average intelligence, without having any major sensory problems (like blindness or hearing impairment), and yet struggles to keep up with people of the same age in learning and regular functioning. Due to such implicit characteristics of learning disabilities, the identification of students with LDs has long been a difficult and time-consuming process. In the United States, the so called "Discrepancy Model" [3], which states that a severe discrepancy between intellectual ability and academic achievement has to exist in one or more of these academic areas: (1) oral expression, (2) listening comprehension (3) written expression (4) basic reading skills (5) reading comprehension (6) mathematics calculation, is one of the commonly adopted criteria to evaluate whether a student is eligible for special education services.

In Taiwan, the diagnosis procedure pretty much follows the "Discrepancy Model". The sources of input parameters required in such prolonged process include information from parents, general education teachers, students' academic performance and a number of standard achievement and IQ tests. To guarantee collection of required information regarding students suspected with LD, usually checklists of some kind are developed to assist parents and regular education teachers. The Learning Characteristics Checklists (LCC), a Taiwan locally developed LD screening checklist [4], is commonly used in most counties of Taiwan. Among the standard tests, the Wechsler Intelligence Scale for Children, Third or Fourth Edition (WISC III or IV) plays the most important role in this LD diagnosis model. WISC-III consists of 13 sub tests [5]. The scores of the sub-tests are then used to derive 3 IQs, which include Full scale IQ (FIQ), Verbal IQ (VIQ), Performance IQ (PIQ), and 4 indexes, which include Verbal Comprehension Index (VCI), Perceptual Organization Index (POI), Freedom from Distractibility Index (FDI), Processing Speed Index (PSI). There are also a number of locally developed standard achievement tests (AT), which typical consist of reading, math, and fields that are related to students' academic achievement.

Diagnosis of students with LDs then involves mainly interpreting the standard test scores and comparing them to the norms that are derived from statistical method. As an example, in case the difference between VIQ and PIQ is greater than 15, representing significant discrepancy between a student's cultural knowledge, verbal ability, etc, and his/her ability in recognizing familiar items, interpreting action as depicted by pictures, etc, is a strong indicator in differentiating between students with or without LD [5]. A number of similar indicators together with the students' academic records and descriptive data (if there is any) are then used as the basis for the final decision. Confirmed possible LD students are then evaluated for one year before admitting to special education. However, it is important to note that a previous study reveals that the certainty in predicting whether a student is having a LD using each one of the currently available predictors is in fact less than 50% [6].

The above identification procedure involves extensive manpower and resources. Furthermore, a lack of nationally

 ISBN: 978-1-61208-290-5

regulated standard for the LD diagnosis procedure and criteria result in possible variations on the outcomes of diagnosis. In some cases, the difference can be quite significant [7].

With the advance in artificial intelligence (AI) and its successful applications to various classification problems, it is interesting to investigate how these AI-based techniques perform in identifying students with LDs. In our previous study, we have shown that ANN classifier does well in positively identifying students with LDs [7]. In subsequent studies, we combined various feature selection techniques and genetic-based parameters optimization with the ANN classifier, which further improves the overall identification accuracy [8]. However, although ANN-based classifier performs well in LD diagnosis problem, the procedure is computation-intensive and may take quite a while to process. Accordingly, multi-threaded programming and grid-based parallel computing (a parallel distributed genetic algorithm based implementation, will be referred to as PDGA hereafter) have been used to speedup the ANN model training and validation [9, 10, 11].

In this paper, we still focus on the ANN classification model and work on porting the GA-based ANN classifier to the MapReduce programming model. To fit into the new programming model, we have done a number of modifications of the PDGA procedure. The rest of the paper is organized as follows. Section 2 briefly describes the history of applying AI techniques to special education and gives a short introduction to Hadoop related terms that are used in our implementation. Sections 3 and 4 present our experiment settings, design and corresponding results. Finally, Section 5 gives a brief summary of the paper and lists issues that deserve further investigation.

II. RELATED WORK

Artificial intelligence techniques have long been applied to special education. However, most attempts occurred more than one or two decades ago and mainly focused on using the expert systems to assist special education in various ways [7]. There were also numerous classification techniques other than neural networks that were developed and widely used in various applications [12]. Among all the classification techniques, artificial neural networks (ANN) have received lots of attentions due to their demonstrated performance and have gained wide acceptance [13].

An artificial neural network is a mathematical representation that is inspired by the way the brain processes information. Many types of ANN models have been suggested in literature, with the most popular one for classification being the multilayer perceptron (MLP) with back propagation. The goal of this type of network is to create a model that correctly maps the input to the output using historical data so that the model can then be used to predict the outcome when the desired output is unknown. MLP with back propagation is typically composed of an input layer, one or more hidden layers and an output layer, each consisting of several neurons. Each neuron processes its inputs and generates one output value that is transmitted to the neurons in the subsequent layer. Fig. 1 provides an example of an MLP with one hidden layer and one output neuron.

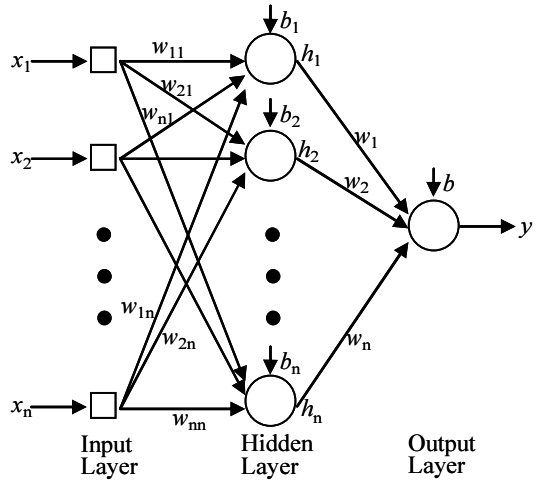


Figure 1. MLP with one hidden layer.

The output of i-th hidden neuron is computed by processing the weighted inputs and its bias term b_i as follows:

$$h_i = f^h\left(b_i + \sum_{j=1}^{n} w_{ij} x_j\right) \quad (2)$$

where w_{ij} denotes the weight connecting input x_j to hidden unit h_i. Similarly, the output of the output layer is computed as follows:

$$y = f^{output}\left(b + \sum_{j=1}^{n} w_j x_j\right) \quad (3)$$

with n being the number of hidden neurons and w_j represents the weight connecting hidden unit j to the output neuron. A threshold function is then applied to map the network output y to a classification label. The transfer functions f^h and f^{output} allow the network to model non-linear relationships in the data. Also note that the number of hidden layer nodes does not need to be the same as the number of input nodes.

The training of a neural network is the process of presenting the network with sample data and modifying the weights to approximate the desired function. In particular, an epoch indicates one iteration through the process of providing the network with a sample input and updating the network's weights. Let N_i, N_h and N_o respectively represent input feature size, number of hidden and output nodes, the total order of complexity is then $O(N_i \times N_h \times N_o + N_h \times N_o)$ for one epoch [14]. Since a typical ANN training process usually takes 500 epochs, the computation complexity for training of an ANN model is roughly equal to $500 \times N \times O(N_i \times N_h \times N_o + N_h \times N_o)$, where N represents the size of input samples for training.

In the field of special education, ANN has been used in a number of applications [7]. To improve the ANN classification accuracy, genetic algorithms have been used in

 ISBN: 978-1-61208-290-5

the training and constructing of ANN model [15]. However, the GA optimization procedure may require numerous applications of the above ANN training process (depending on the number of chromosomes and evolution generations), and thus usually takes quite a long time to process [7]. Accordingly, researches have been applying parallel processing, which may provide affordable computational power, to speedup the time-consuming process [16]. For network connected cluster or grid environment, message passing interface (MPI) is usually used to coordinate computing nodes for completing a common task. On the other hand, to take full advantage of the currently available multi-core processor technology, OpenMP may be used explicitly to direct multi-threaded, shared memory parallelism [9].

With the advance of the cloud computing, a number of distributed computational models have also been developed. Among them, the MapReduce, together with GFS and GigTable were developed by Google in 2003. MapReduce is a programming model for large-scale data processing problems, which may separate the original problem from the details of parallelization. However, other than the related documents and algorithms, Google did not release their source codes. Fortunately, Hadoop, developed by Apache foundation that originally includes HDFS, HBase and MapReduce, is an open-source alternative for Google's implementation [17].

HDFS (Hadoop Distributed File System) is designed to operate upon low-cost hardware with high fault-tolerance and provide high throughput access to applications that have massive data sets. An HDFS cluster operates in a master-slave setup consisting of a name-node (master) and a varying number of data-nodes (slaves). The name-node maintains the metadata for all the files and directories in the file system. It also knows the data-nodes on which all the blocks for a given file are located [17].

MapReduce, operating upon the HDFS, is a distributed programming model that may work on a cluster of tremendous computational nodes and is suitable for processing problems with massive data sets. In MapReduce programming model, a computation is specified by two functions: Map and Reduce. The underlying MapReduce library then proceeds to parallelize the computation, while hiding issues such as data distribution, load balancing and fault tolerance from the programmers. Accordingly, MapReduce programmers may thus be able to concentrate on the programming logic in solving the problems.

A MapReduce job, which consists of input data, MapReduce program, and configuration information, is divided into map and reduce tasks. The job-tracker and a varying number of task-trackers control the job execution process, with the job-tracker coordinating all the jobs on the system and the task-trackers running tasks and sends job progress to the job-tracker [17]. The configuration of various roles in a Hadoop cluster environment can be shown in Fig. 2. As can be seen, the master can be a job-tracker / name-node and a task-tracker / data-node at the same time, while a slave can only be a task-tracker and a data-node.

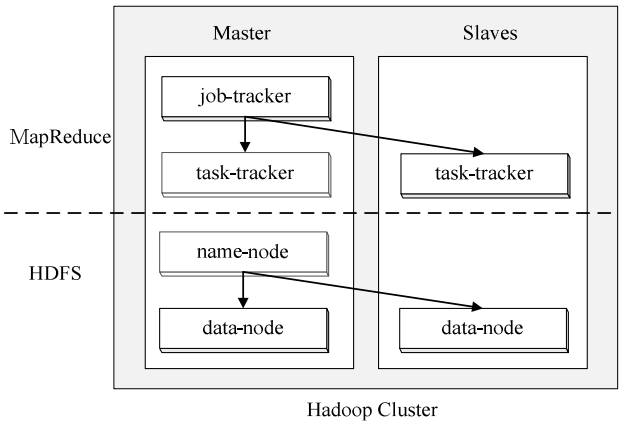


Figure 2. Various roles of Hadoop cluster nodes (revised from [18]).

In this study, we will work on porting the GA-based ANN classifier for LD identification [9] to the emerging cloud computing paradigm.

III. ENVIRONMENT SETUP AND IMPLEMENTATION ISSUES

A virtualized 12-node mini-cloud environment, established on top of 2 multi-cores PCs running Ubuntu server, is set up for the experiment. The hardware details of the PCs and the mini-cloud setup are shown in Table I and Fig. 3. Note, virtualization (through kernel-based virtual machine: KVM) is adopted in this study so that we may take full advantage of the current multi-core CPU technology.

TABLE I. HARDWARE DETAILS OF THE PCS IN OUR STUDY

	CPU	No. of cores	Memory
PC 0	Intel (R) Core (TM) i7 (2.7 GHz)	4 physical cores (8 logical cores)	12 GB
PC 1	AMD Phenom (TM) II (3.3 GHz)	6 physical cores	8 GB

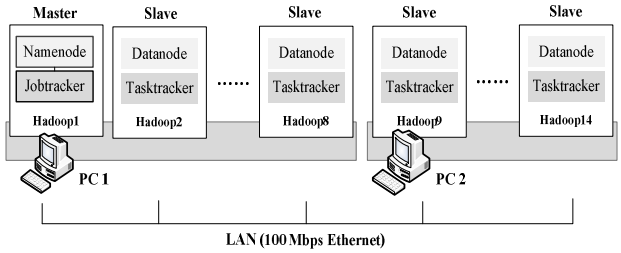


Figure 3. The mini-cloud setup in our study.

To map the regular genetic algorithm to the MapReduce model, we re-arrange the order of the GA procedure as shown in Fig. 4. The most computation-intensive step, which would be the fitness function calculation (ANN model construction and validation), is implemented in the Map stage, while the other GA processes such as selection, cross-over, and mutation are organized in the Reduce stage. Note there is only one Reducer in our implementation, which means only the most computation-intensive fitness function is parallelized while the GA processes are executed

sequentially. Although this may somewhat degrade the overall performance (in terms of execution time), yet the implementation is much simpler and we may also avoid the GA procedure converging to some local maxima when each reduce task is distributed with too few chromosomes in a multiple Reducers setup [10]. Furthermore, as our input data is much smaller than the HDFS block size (64 MB), we manually split the input data for each map task to avoid potential overhead in managing the splits and map task creation when it is done by Hadoop [17].

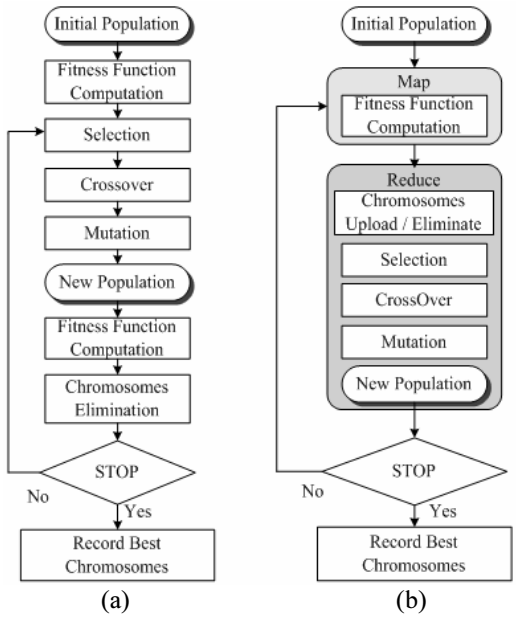


Figure 4. (a) a regular GA operation process that we adopted in our earlier PDGA implementation [9, 10, 11], and (b) its mapping to the corresponding MapReduce programming model.

In addition, as shown in Fig. 5, in each generation the reduce task would preserve at most *N* best chromosomes in the HDFS.

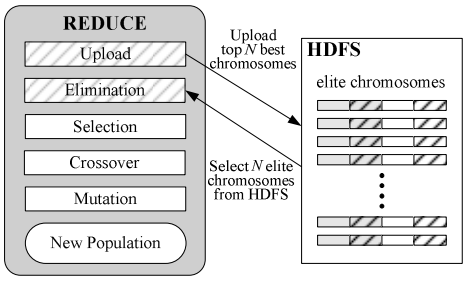


Figure 5. Elite chromosomes preservation and distribution.

Note, these *N* chromosomes also have to be better, in terms of accuracy, than a threshold value (which would be the average of all the elite chromosomes stored in HDFS) to be preserved. Those accumulated elite chromosomes may later be randomly selected to replace the *N* worst chromosomes in consecutive generations. In all of our experiments in this study, *N* is set to 5.

IV. EXPERIMENT DESIGNS AND RESULTS

Our objectives in this study are two-fold: (1) to gain some experience in a mini-cloud environment, and hopefully this may be extended to future application with more input and in a larger scale cloud environment setup, (2) to evaluate how the parallel genetic algorithm performs in constructing the ANN-based LD identification model with the MapReduce programming model as compared to implementations using multi-threaded APIs (OpenMP) and grid-based distributed computing.

The data sets used in this study are summarized in Table II, which together with the corresponding pre-processing (such as normalization and feature selection) are exactly the same as those used in [9].

TABLE II. DATA SETS AND THEIR FEATURES USED IN THIS STUDY

	sample size	number of features
data set 1	652	7
data set 2	125	7
data set 3	159	10

To fulfill the above mentioned objectives, we have design and conducted three experiments. The ANN code (fitness function computation, adopting five-fold cross validation and 500 epochs in each ANN training) is exactly the one used in [9] (in C language), and is invoked by the Map tasks (implemented with Java language) through external procedure call. Three parameters of the ANN classifier (number of hidden nodes, learning rate and momentum), together with random number seeds, which might affect the initial weights and bias of neural network, are encoded into the chromosomes. For genetic algorithm, real-value encoding is adopted with the crossover rate, mutation rate and number of generation set at 0.8, 0.1 and 50, respectively. Furthermore, accuracy in classification is used to evaluate the fitness of populations. A performance index: correct identification rate (CIR) is defined to evaluate the experiment outcomes, as listed in equation 1 below.

$$CIR = \frac{(\text{number of correct LD and non - LD identification})}{(\text{total number of cases})} \quad (1)$$

In the first experiment, we evaluate our MapReduce implementation of the GA-based ANN classifier in terms of CIR and execution time by fixing the population size (number of chromosomes) assigned to each map task to 20 in the PDGA-based ANN classifier, while varying the number of computing nodes (1, 2, 4, 8, and 12, respectively). Accordingly, the overall population size also varies between 20, 40, 80, 160, and 240, respectively. In the second experiment, we fix the overall population size to 200, while varying the number of computing nodes (1, 2, 4, 8, and 12, respectively). In other words, the overall population is evenly distributed to each map task (in case of 12 nodes scenario, each node is assigned 17 chromosomes). The results of the two experiments are shown in Tables III and IV, with all numbers as averaged over twenty consecutive runs.

In general, according to Table III, the CIR improves as the overall population size increases. From Table IV, when

 ISBN: 978-1-61208-290-5

adding more computing nodes (and thus reducing population size assigned to each individual node), it is possible to achieve higher CIR and lesser execution time at the same time. The above two findings are reasonable and consistent with our previous studies [9, 10].

TABLE III. PERFORMANCE COMPARISON BY FIXING POPULATION AT EACH NODE TO 20 AND VARYING COMPUTATIONAL NODES (ALL TIME IN SECONDS)

data set / slave node	1		2		3	
	CIR	*execution time*	*CIR*	*execution time*	*CIR*	*execution time*
1	87.9%	4048	84.7%	2390	86.2%	3655
2	87.9%	3689	84.9%	1998	86.4%	3401
4	87.9%	3624	85.4%	2275	86.4%	3594
8	87.9%	4431	85.8%	2584	86.6%	4289
12	87.9%	5145	86.2%	3641	86.9%	4831

TABLE IV. PERFORMANCE COMPARISON BY FIXING THE TOTAL POPULATION TO 200 AND VARYING THE COMPUTATIONAL NODES (ALL TIME IN SECONDS)

data set / slave node	1		2		3	
	CIR	*execution time*	*CIR*	*execution time*	*CIR*	*execution time*
1	88.0%	20368	85.9%	8170	86.3%	16521
2	88.0%	9781	85.8%	4116	86.8%	8807
4	88.0%	6456	85.6%	3046	86.9%	6545
8	**88.1%**	5293	85.8%	2977	86.5%	4386
12	88.0%	4378	85.7%	3407	86.9%	5035

However, in [10], we notice that in the later case (experiment 2) there may be a limit on the trend. It appears the sub-population assigned to each node has to be at least 20 to avoid the possibility that evolutionary process contains too few chromosomes and potentially causes the GA optimization process to be trapped into some local maximum. But we do not see this obvious trend in Table IV. One possible reason may be the sub-population size (17) in the 12-node scenario is very close to the above mentioned threshold (20). However, it is more likely due to our non-parallelized implementation of the Reduce stage where the GA procedure proceeds. In other words, no matter how many nodes are involved, all 200 chromosomes are taking part in the evolutionary phase in one node. Furthermore, we need to note that 88.1% (in Table IV) is the best (average) CIR we have achieved so far with data set 1.

In the last experiment, we compare our MapReduce implementation of the GA-based ANN classifier and the grid and OpenMP implementations in terms of CIR and execution time by varying the population size in a fixed 7-node mini-cloud environment. By OpenMP, we mean OpenMP APIs are used to multi-thread the most time-consuming ANN model constructions and verifications in our case. A simple static scheduling that evenly assigns population to the available threads (cores) is adopted. The outcomes are shown in Table V, again with all numbers as averaged over twenty consecutive runs. Note that all three parallel computing environments are built upon PC0 as listed in Table I so that the performance comparison can be meaningful. In addition, the outcomes of the sequential version (depicted as NA) of our ANN classifier implementation are also shown and used as the baseline for comparison.

TABLE V. PERFORMANCE COMPARISON ON VARYING THE POPULATION IN THE 7-NODE SETUP (1 MASTER + 6 SLAVES, ALL TIME IN SECONDS, NA, MR, GRID AND OMP REPRESENT SEQUENTIAL, MAPREDUCE, GRID COMPUTING AND OPENMP IMPLEMENTATIONS, RESPECTIVELY)

data set / population		1		2		3	
		CIR	*execution time*	*CIR*	*execution time*	*CIR*	*execution time*
100	*NA*	87.5%	6302	84.7%	1998	86.9%	4001
	MR	**87.9%**	3935	85.3%	2225	86.6%	3581
	Grid	87.4%	1991	**85.7%**	798	**87.2%**	1122
	Omp	87.3%	1450	84.6%	435	86.7%	847
200	*NA*	87.6%	13460	84.8%	4545	87.0%	7450
	MR	**88.0%**	5600	**86.0%**	2763	86.9%	4974
	Grid	87.3%	3775	85.7%	1513	**87.7%**	2100
	Omp	87.6%	2562	85.0%	908	86.8%	1544
300	*MR*	**88.1%**	6809	86.4%	3128	87.2%	5889

Only MapReduce implementation outcomes are available in the case of 300 population size.

According to Table V, it seems distributed implementations (either MapReduce or grid computing) perform somewhat better in terms of CIR. But when it comes to execution time, the OpenMP version of the PDGA performs the best (with speedup between 4.35 and 5.25), and the grid implementation stands second (with speedup between 2.50 and 3.57), and the MapReduce implementation falls far behind (with speedup between 0.90 and 2.40). The primary cause may be attributed to the sequential operation in the Reduce stage (the GA procedure), which in our measure may take between 25% (population=300) to 50% (population=100) of the overall execution time. Accordingly, the parallelization of the Reduce stage would be our first priority in future research.

In addition, we also note that CPU usage jumps from 23% with sequential implementation to 92% with multi-thread implementation using OpenMP APIs. In cases of MapReduce and grid computing implementations, the CPU usage can be as high as 100%. Apparently, the computing power of the underlying multi-core CPUs has indeed been fully utilized. However, considering the speedup depicted above, there may be quite a lot of work to do in reducing overhead associated with the MapReduce and grid implementations (especially with the former one), which would be another focus of our future study.

V. SUMMARY AND FUTURE WORK

In this study, we modify our grid-based PDGA implementation of the ANN classifier for identifying students with learning disabilities to the MapReduce distributed programming model. Compared with the grid computing model, MapReduce has the advantage of hiding

 ISBN: 978-1-61208-290-5

the underlying hardware details and thus allow the programmers to be able to concentrate on the programming logic in solving the problems. The preliminary results show that in 50% of cases, the MapReduce implementation may achieve the best CIR when compared to the other parallel programming models. However, in terms of execution time, the MapReduce model does not show significant breakthrough. But we do see the potential of the MapReduce model in our future applications. For example, increase the population size, which may easily be extended by simply adding more nodes to the Hadoop-based cloud environment, seems to be a good direction to optimize the ANN LD classification model. In addition, more diagnosis data for students with LDs will be collected so that we may explore the processing power of MapReduce upon massive data sets. Finally, a more sophisticated parallelized GA procedure in the Reduce stage is also under development.

ACKNOWLEDGMENT

This work was supported in part by the National Science Council of Taiwan, R.O.C. under Grant NSC 100-2511-S-018-011-MY2.

REFERENCES

[1] S. A. Kirk, "Behavioral diagnosis and remediation of learning disabilities," Proc. of the Conference on the Exploration into the Problems of the Perceptually Handicapped Child, 1963, pp.1-7.

[2] J. M. Fletcher, W. A. Coulter, D. J. Reschly, and S. Vaughn, "Alternative approach to the definition and identification of learning disabilities: some questions and answers," Annals of Dyslexia, vol. 54, no. 2, 2004, pp. 304-331.

[3] J. Schrag, "Discrepancy approaches for identifying learning disabilities," http://www.specialed.us/discoveridea/topdocs/nasdse/discld.pdf, retrieved: June, 2013.

[4] Y.-R. Meng and L.-R. Chen, "On discussing the differences about the learning characteristics of LD," Bulletin of Special Education, vol. 233, 2002, pp. 75-93. (in Chinese)

[5] C. L. Nicholson and C. L. Alcorn, "Interpretation of the WISC-III and its subtests," Paper presented at the 25th Annual Meeting of the National Association of School Psychologists, Washington, DC, 1993.

[6] T.-S. Huang, "A Study on the characteristics of WISC-III for students with learning disabilities," Master thesis, Graduate Institute of Special Education, National HsinChu University of Education, Hsinchu, Taiwan. (in Chinese)

[7] T.-K. Wu, S.-C. Huang, and Y.-R. Meng, "Evaluation of ANN and SVM classifiers as predictors to the diagnosis of students with learning disabilities," Expert Systems with Applications, vol. 34, no. 3, April 2008, pp. 1846-1856.

[8] T.-K. Wu, S.-C. Huang, and Y.-R. Meng, "Effects of feature selection on the identification of students with learning disabilities using ANN," Lecture Notes in Computer Science, Springer Berlin/Heidelberg, vol. 4221, 2006, pp. 565 – 574.

[9] T.-K. Wu, S.-C. Huang, Y.-R. Meng, Y.-L. Lin, and H. Chang, "On the parallelization and optimization of the genetic-based ANN classifier for the diagnosis of students with learning disabilities," Proc. 2010 IEEE Conference on Systems, Man and Cybernetics, 2010, pp. 4263-4269.

[10] T.-K. Wu, S.-C. Huang, Y.-R. Meng, and T.-H. Wu, "Experiences on constructing neural network based learning disabilities identification model with the Amazon elastic compute cloud," Proc. 2012 International Conference on Internet Study, 2012.

[11] K. Kazunori, M. Hiroshi, and I. Masaaki, "Asynchronous Parallel Distributed GA using Elite Server," Proc. 2003 congress on evolutionary computation, 2003, vol. 4, pp. 2603-2610.

[12] B. Baesens, T. Van Gestel, S. Viaene, M. Stepanova, J. Suykens, and J. Vanthienen, "Benchmarking state-of-the-art classification algorithms for credit scoring," Journal of the Operational Research Society, vol. 54, 2003, pp. 627–635.

[13] C. M. Bishop, Neural Networks for Pattern Recognition, Oxford University Press, Oxford, UK, 1995.

[14] E. Istook and T. Martinez, "Improved backpropagation learning in neural networks with windowed momentum, International journal of neural systems," vol. 12, no.3 & 4, 2002, pp. 303-318.

[15] E. Cantú-Paz and C. Kamath, "An empirical comparison of combinations of evolutionary algorithms and neural networks for classification problems," IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics, vol. 35, no. 5, 2005, pp. 915-927.

[16] N. Sakamoto, K. Ozawa, and T. Niimura, "Grid computing solutions for artificial neural network-based electricity market forecasts," Proc. 2006 International Joint Conference on Neural Networks, 2006, pp. 4382-4386.

[17] T. White, Hadoop: The Definitive Guide. O' Reilly Media, Inc.

[18] Y. Wang and W. Chen, "Introduction to the Hadoop distributed file system," http://trac.nchc.org.tw/cloud/raw-attachment/wiki/NCHCCloudCourse090331/3.ppt, retrieved: June, 2013.

 ISBN: 978-1-61208-290-5

Principles and State-of-the-Art of Engineering Optimization Techniques

Ning Xiong, Miguel León Ortiz
School of Innovation, Design and Engineering
Mälardalen University
Västerås, Sweden
E-mail: ning.xiong@mdh.se, miguel.leonortiz@mdh.se

***Abstract*—This paper gives a survey of the principles and the state-of-the-art of engineering optimization techniques. Both the classic and emerging approaches to nonlinear optimization problems are reviewed and analyzed. All the techniques are discussed in two basic types: point-based transition and population-based transition, depending on whether a single point or multiple points are generated as new approximate solution(s) in each step. We also consider multi-objective tasks as new application trend and point out the strong potential of population-based methods to tackle multiple objectives simultaneously.**

Keywords-optimization techniques; point-based optimization; population-based optimization

I. INTRODUCTION

Nowadays, optimization has become an important issue in industrial design and product development [1]. It is necessary to enhance system performance whereas reduce product cost to meet challenges in the competitive market. From engineering perspective, optimization means adjusting or fine tuning system designs in terms of one or more performance factors. This is not a trivial task, in particular when the problem space is complex and of high-dimensionality. Application of suitable optimization techniques has shown its benefit in supporting human designers to acquire optimal or near optimal solutions within a short design time.

Generally, an engineering optimization problem can be formulated as:

$$\text{minimize} \quad f(X) = f(x_1, x_2, \cdots, x_n)$$

$$\text{subject to} \quad g_i(x_1, x_2, \cdots x_n) \le 0, \quad i = 1 \cdots m$$

where (x_1, x_2, …x_n) is the vector of design variables, $g_i(i=1 \ldots m)$ denote constraint functions that define the region of feasible solutions in the problem space, and function $f(X)$ provides objective values for vectors of variables representing alternative designs. The set of design variables x_i can take continuous or discrete values or a mixture of both depending on specific problems. Besides, the above statement is generic since maximization of a certain function is equivalent to minimization of the minus of it.

The optimization techniques can be divided into two basic categories: linear programming [2] and non-linear programming [3], [4]. The former is applied to optimization problems that have linear objective and constraint functions. Important progresses in this area include the polynomial-time ellipsoid algorithm [5] and the interior point algorithm [6], both were proposed to reduce time complexity and to allow for extremely efficient problem handling in the optimization procedure. At present, linear programming has been advanced to a soundly founded discipline and widely used technology for linear optimization problems. The second category of optimization is called nonlinear programming, which refers to the consortium of methods and approaches that are designed to deal with problems with nonlinear objective or constraint functions [7]. Nonlinearity is a very common property for many engineering optimization problems, and solving such problems often presents a challenge due to the high complexity, high dimensionality and multi-modality of the problem space. Although many techniques for nonlinear programming have been developed, there are always pros and cons for them and no single method can more competently solve all kinds of problems than others.

This paper focuses on the study of nonlinear optimization techniques. Special emphasis is made on presenting the general principle and ideas of how to reach the optimum rather than the details of computational procedures. Both the classic and emerging approaches to nonlinear optimization problems are reviewed and analyzed. We also consider multi-objective tasks as new application trend when discussing the potential capability of optimization methods.

The organization of this paper is as follows. Section II highlights the basic idea and principle for general nonlinear optimization problems. The review of concrete approaches for optimization is given in Sections III and IV, respectively. The type of approaches called **point-based transition** is discussed in Section III, and the type of approaches called **population-based transition** is addressed in Section IV. Finally, Section V provides concluding remarks and discussion.

II. GENERAL PRINCIPLE OF OPTIMIZATION

Mathematically, it is well known that an optimum of a nonlinear function $f(x_1, x_2, \ldots x_n)$ must be some point at which the partial derivatives of the function with respect to all variables are equal to zero, i.e.,

$$\frac{\partial f}{\partial x_i} = 0, \qquad i = 1, 2, \cdots, n \tag{1}$$

A solution satisfying all the equations in (1) is called a stationary point of function f. Further, the stationary point is

 ISBN: 978-1-61208-290-5

a minimum solution if the Hessian matrix of the second-order derivatives, as defined in (2), is positive definite.

$$H = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix} \quad (2)$$

The above principle suggests a simple procedure to obtain exact solution of an optimization problem. It is done by finding all stationary points of the objective function and then examining the property of the Hessian matrices of these points. The global optimum solution is selected from those stationary points for which the Hessian matrices are positive definite.

Unfortunately, the approach to exact solutions of optimization is rarely applicable in engineering practice. The main reason lies in the difficulty of acquiring the derivative information analytically. In many applications, only concrete objective values of individual designs are calculable through specific calculations such as simulation. But the explicit expression of the objective function is not available, not to mention the analytic formulation of the partial derivative functions. It follows that we are unable to construct the equations as formulated in (1) for determining the stationary points of the objective function.

Numerical approaches present a pragmatic alternative to solve engineering optimization problems. The main idea is to create arbitrary initial approximate(s) to the problem and then improve them progressively. The whole procedure consists of a number of iterations. In each iteration, new approximate(s) are created from the old one(s) as more promising solution(s). Depending on the number of approximates generated at a single step, two types of numerical approaches (point-based and population-based transitions) can be defined and explained as follows.

With point-based transitions, only one point as new approximate is generated and evaluated in each of the iterations. The new point is made as transition from an old one with expected better performance. The general form of such a transition can be expressed as

$$X_{i+1} = X_i + h_i S_i \quad (3)$$

where X_{i+1} and X_i denote the new and old approximates respectively, h_i decides the length of transition, and S_i is a vector determining the direction of the move from X_i. There are many different methods to determine the direction vector S_i in the literature. Some use merely values of the objective function while others require partial derivative information in addition to the objective values.

Sometimes, it is beneficial to apply point-based transition methods in combination with random sampling to increase their global search ability and thereby reducing the risk of getting stuck into local optima. For instance, the initial approximate for iteration can more favorably be decided by resorting to a random scheme [8], which generates a set of uniformly distributed points in the region of feasible solutions. We then select the sample solution that receives the best objective value as the starting point of search. The other possibility is to follow the multi-start strategy [9] when doing optimization with point-based transitions. This means that we run the optimization algorithm multiple times and every time a sample solution is selected randomly as the starting point. The best solution found from individual runs is treated as the final solution of the global optimum.

Population-based transition starts from an initial population of feasible solutions. Then it undergoes an iterative procedure in which new populations are successively created from old populations to reach progressively refined approximates to the problem. As many points in the space are explored simultaneously, population-based transition is superior to point-based transition in the global search ability; hence it has less likelihood of ending with a local minimum. Many biologically inspired optimization techniques rely on transitions of populations, such as genetic algorithms, memetic algorithms, differential evolution, particle swarm optimization, as well as ant colony optimization, which will be reviewed in Section IV.

III. OPTIMIZATION WITH POINT-BASED TRANSITION

The approaches of this type explore the problem space via transition from one feasible solution to another. The transition procedure is controlled by either deterministic or probabilistic rules. Six well known approaches in this category will be surveyed here.

A. Hill-Climbing

Hill-climbing [10] is the simplest numerical approach for optimization. It starts by creating an arbitrary solution (approximate) to the problem and then it evaluates all the neighbors of the current solution. If the best neighbor has a lower objective value than the current solution, the current one is replaced by that neighbor and the search moves on to the next iteration, otherwise the search is terminated.

Hill-climbing is a local search and can only be applied in discrete spaces as it implicitly assumes a finite number of feasible neighbors at every point. The advantages of hill-climbing lie in its simplicity and high efficiency. It has been widely used to solve many machine learning and technical optimization problems (e.g., [11], [12]). Hill-climbing is particularly recommended when there is limited time for search; for example, for real-time systems.

B. Gradient Descent

Gradient descent [13] aims to solve continuous optimization problems. It has very similar idea to that of hill-climbing. The only difference between both methods lies in the way to determine the best successor solution from a current one. Since it is not possible to evaluate every successor in the continuous space, the gradient information is utilized to identify the direction of move to reduce the objective function most quickly. Hence, the normalized direction vector of the move can be written in (4). The length

 ISBN: 978-1-61208-290-5

of move along S_i can be determined by solving a one-dimensional optimization problem. The golden section method is often used in gradient descent to find the optimal size of transition at each step.

$$S_i = \frac{\left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \cdots, \frac{\partial f}{\partial x_n}\right)}{\sqrt{\sum_{i=1}^{n}\left(\frac{\partial f}{\partial x_i}\right)^2}} \quad (4)$$

Gradient descent is simple and very useful in solving many optimization problems when partial derivatives of the object function are available. However, as local search scheme, this method cannot guarantee the global optimality of the solutions returned. It should preferably be combined with the multi-start strategy to increase the chance of finding the global minimum. The other weakness with gradient descent is that, when the current solution gets close to a minimum solution, the search will become quite inefficient due to the decreasing lengths of the moves.

C. Newton's Method

Newton's method [14] attempts to improve the speed of convergence of gradient descent in the vicinity of a minimum solution. According to Taylor's expansion, the objective function near a minimum X^* can be expressed by an approximate form as:

$$f(X) = f(X^* + \Delta X) \approx f(X^*) + (\Delta X)^T g + \frac{1}{2}(\Delta X)^T H(\Delta X) \quad (5)$$

where g is the vector of the first-order partial derivatives of the objective function and H is the Hessian matrix of the second-order partial derivatives of the objective function. For all the first-order derivatives at the optimum X^* are zero, Eq.(5) is simply rewritten as:

$$f(X) \approx f(X^*) + \frac{1}{2}(\Delta X)^T H(\Delta X) \quad (6)$$

From (6), it can be seen that the objective function is approximately quadratic in the vicinity of X^*. A quadratic function has the following property:

$$g = H(\Delta X) \quad (7)$$

where g is the vector of partial derivatives evaluated at the current point, and H is the Hessian matrix which is constant for a quadratic function. Using the property in (7) enables us to obtain the minimum solution X^* from a nearby point X in terms of the following transition rule:

$$X^* \approx X - H^{-1} g \quad (8)$$

This transition rule indicates that a single move suffices to reach the minimum X^* when the current solution is nearby. This shows a substantial improvement of the late convergence speed compared with that of gradient descent.

Nevertheless, it has to be noticed that globally the objective function is not quadratic. Hence, an iterative procedure is needed to generate a sequence of moves for progressive refinement. At iteration i, we first calculate g and H at the current point X_i, and then, we use the Newton's method to create the next refined solution as

$$X_{i+1} \approx X_i - H_i^{-1} g_i \quad (9)$$

Generally, the Newton's method stated above is still a local approach and it has two drawbacks. Firstly, it requires heavy computation with the Hessian matrix of second-order derivatives and its inverse. Secondly, the method is only efficient in the neighborhood of an optimum, but away from the optimum it may progress very slowly and even diverge. So, our suggestion is not employing the Newton's method alone, but in combination with some other optimization technique and using it at the final stage.

D. Tabu Search

Tabu search [15] [16] is a metaheuristic local search algorithm to solve discrete optimization problems. It can be considered as extension of the hill-climbing search in the two aspects as follows. Firstl, there is added driving force to enforce the local minimization procedure out of a local minimum. Secondly, various memory structures are used to store historical information which is then utilized to guide the further exploration of new solutions. For instance, the tabu list is introduced as short term memory to save recently visited solutions to prevent cyclic behaviors during the search. Intermediate and long term memory is used to intensify and diversify the search to ensure adequate exploration of the problem space.

The search starts from an arbitrary point as the current solution. All the solutions in its neighborhood that are not in the tabu list or satisfy the aspiration level are successor solutions and their objective values are calculated. Then the move is made to the best successor according to the objective values, and the tabu list is updated accordingly. Both uphill and downhill moves are allowed here to give chance to escape from a local minimum. This process is repeated in a number of iterations until the termination condition is satisfied.

It is useful to apply tabu search in many practical scenarios [17] [18], mainly in combinatorial problems. A main limitation with this technique is that it requires considerable memory resources to store historical information. Besides, domain specific knowledge is needed to design suitable aspiration criteria.

E. Simulated Annealing

Simulated annealing [19] [20] is a stochastic and meta-heuristic algorithm for solving global optimization problems. It is inspired by the physical principle of annealing used in material engineering. In an annealing process, the solid is first heated to a high temperature, causing atoms to move away from their initial positions. When the material cools down slowly, the atoms adjust themselves into a new thermal equilibrium that corresponds to a minimum energy state.

 ISBN: 978-1-61208-290-5

The algorithm is iterative and the main idea is to randomly select a new solution in the neighborhood of the current solution at every step. The difference of objective values, ΔE, between the new and current solutions is calculated as analogy to the change of energy. If the new solution is better than the current one (ΔE<0), the current solution is replaced by the new one. In case when the new solution is worse (ΔE>0), there is still a chance to move to it. The probability of this move is given by the Boltzmann probability function:

$$P(\Delta E) = \exp\left(-\frac{\Delta E}{T}\right) \quad (10)$$

The parameter T in (10) is the temperature used during the search. In the early iterations, the temperature is high, which results in high probability of moving into inferior points and thereby avoiding local minima. Contrarily, during late stages, the temperature is reduced to such a level that gives little chance for accepting worse solutions and consequently the search will finally converge.

The merit with simulated annealing is that it does not require the objective function to be continuous and differentiable, and it can handle both continuous and discrete optimization problems. But, proper parameter settings with this method are not difficult.

F. Simplex Method

A simplex is a geometric object consisting of n+1 points (vertices) in the n-dimensional spaces. Every vertex of the simplex corresponds to a feasible solution to the problem. The initial simplex can be generated randomly. The main idea is to move the simplex iteratively and every time replacing the worst vertex of it with a new better point. The search terminates when the standard deviation of the objective values of the n+1 vertices of the simplex is lower than a specified value.

According to the simplex method by Nelder and Mead [21], the new better points for replacement are generated through the operations such as reflection, expansion, and contraction. The worst vertex is first reflected through the centroid of the remaining points of the simplex. If the reflection produces a better point, expansion is done to see whether the objective function can be reduced further via moving in the same direction. Otherwise, if the reflected point is not satisfactory, contraction is performed via generation of a point between the worst vertex and the centroid for possible replacement.

The main advantage of the simplex method is that it is a global optimization technique and it does not require any derivative information of the objective function. The method is robust and efficient with a small number of design variables but it does not scale well up to high-dimensional problems. As noted in [7], the efficiency of simplex diminishes when the design variables are more than five.

IV. OPTIMIZATION WITH POPULATION-BASED TRANSITION

The approaches of this type explore the problem space via transition from one population of feasible solutions to another. They are biologically inspired techniques and probabilistic rules are used to create new solutions from old ones. Five well known approaches in this category will be reviewed here.

A. Genetic Algorithms

Genetic algorithms (GAs) are stochastic optimization algorithms that emulate the mechanics of natural evolution [22]. They are attractive to be applied in engineering optimization tasks due to the two following reasons. First, a GA evaluates many points in the search space simultaneously, as opposed to a single point, thus reducing the chance of converging to the local optimum. Second, a GA uses only values of objective functions; therefore they do not require the search space to be differentiable or continuous.

Essentially, a GA is an iterative procedure maintaining a constant population size. An individual in the population encodes a possible solution to the problem with a string analogous to a chromosome in nature. At each step of iteration, new individuals are created via applying genetic operators on selected parents, and subsequently some of the old, weak individuals are replaced by new strong ones. In this manner, the performance of the population will be gradually improved in the evolutionary process.

A classical GA works with binary code, i.e., individuals in the population are represented by binary strings. However, binary coding would not be the most appropriate choice in applications to optimization problems with continuous spaces. One reason lies in the matter of resolution, i.e., a binary string is inherently related to some loss of precision for representing the continuous value of a variable. The other reason is the extra job of decoding that is needed when doing fitness evaluation for a binary string in the population.

The other alternative is to directly adopt arrays of real numbers as population individuals. Real-coded GAs have been studied by many researchers and nowadays become a popular, extended version of GAs for solving real-valued optimization problems. The interesting features of real-coded GAs together with their used mechanisms and genetic operators haven been carefully discussed in [23] and [24]. In [25], real-coded GA was used to optimize similarity models for case-based reasoning.

B. Memetic Algorithms

Memetic algorithms (MAs) [26] are population-based metaheuristic search methods inspired by the principle of natural evolution and Dawkin's notion of memes capable of local adaptation. MAs can be considered as enhancement of GAs by embedding local search to allow for self-refinements of individuals. According to the idea of

 ISBN: 978-1-61208-290-5

Lamarckian learning [27], local search can be done on all or part of the population to reach a local optimum or improve the current solution.

As hybridization of GAs and local search, MAs aim to exploit the best search regions gathered by global sampling with GA. Hence, an important demand for MAs is the synergy between the exploration abilities of the GA and the exploitation abilities of the local search. In [28], a local search scheme was combined with the global search capability of evolutionary algorithms for rule extraction. The simplex method by Nelder and Mead was adopted as the local search mechanism in the memetic algorithm [29] for optimal fuzzy controller design. The hierarchical memetic algorithm was proposed in [30] for combined feature selection and similarity modeling in case-based reasoning.

C. Differential Evolution

Differential evolution (DE) [31] is a stochastic and meta-heuristic technique that has been developed for solving optimization problems with real parameters. It provides a powerful tool for searching for optimal solutions in high-dimensional spaces that are nonlinear, non-differentiable, non-continuous, and containing multiple local optima. DE is similar to GAs in the sense that both are evolutionary, population-based algorithms. But DE algorithms differ from GAs in the way evolutionary operators are manipulated to produce new child solutions. The main loop of DE algorithms is briefly explained in the following.

A DE algorithm maintains a population of real-valued parameter vectors and works iteratively. Each iteration starts with mutation, in which three distinct parameter vectors are randomly selected for every population member. The weighted differences between two parameter vectors are added to the third parameter vector to get the perturbed vector. Then, crossover is done to combine the population member and the perturbed vector to yield a new trial vector. Every parameter in the perturbed vector has a certain probability to enter the trial vector. Finally, the trial vector replaces the old population member if it has a lower objective value. A comprehensive review of various DE algorithms together with associated operators is given in [32].

DE attains increasing popularity in engineering applications due to its attractive features such as fewer running parameters to specify, ease in programming, high efficiency, as well as strong global search ability. In [31] and [33], it was indicated that DE algorithms were more efficient and more accurate than several other optimization methods, including controlled random search, simulated annealing and genetic algorithms. A weakness for DE is that there is no theoretic proof for its convergence.

D. Particle Swarm Optimization

Particle swarm optimization (PSO) algorithms [34] mimic the flocking behaviors of animals in their movement. Similar to GAs, PSO algorithms work with a population of particles which represent feasible solutions to the problem. The particles move around in the search space to improve their fitness (objective values), iteratively. The movement of each particle is determined in terms of both its best position in the history and also the best position known so far from all particles. In view of this, the speed of a particle at iteration k+1 is updated as:

$$v_{k+1} = w \cdot v_k + c_1 \cdot r_1 \cdot (P_{pb}^k - X_k) + c_2 \cdot r_2 \cdot (P_{gb}^k - X_k) \qquad (11)$$

In (11), P_{pb} , and P_{gb} , denote, respectively, the best position of the particle and the best known position from all particles, w is the momentum, X_k is the position of the particle at iteration k, r_1 and r_2 are two randomly generated positive numbers, and c_1 and c_2 are the parameters used to balance the individual and social influences.

PSO is simpler than GAs in its nature. Therefore, it incurs lower computational cost in creating a new population from the old one. But, the performance of PSO is heavily dependent on the parameters w, c_1, and c_2 in (11), and proper valuation of such parameters to ensure strong search ability is a crucial task.

E. Ant Colony Optimization

Ant colony optimization (ACO) [35] [36] mimics the behavior of a colony of ants in searching for food. It is a population-based metaheuristic technique used to solve hard combinatorial problems. Prior to applying ACO, the optimization problem has to be transformed into the problem of path finding on a graph. Then a group of ants work collectively to find a shortest path on the graph by pheromone communication during path formation [37].

An ant builds its path incrementally. It starts from a randomly selected vertex and then chooses an edge to go to the next vertex. The choice of an edge is stochastic yet its probability is decided by the pheromone values and heuristic information associated with the edge. The most well known rule for determining the selection probability for edge c_{ij} is given in (12)

$$P(c_{ij}) = \frac{\tau_{ij}^{\alpha} \eta_{ij}^{\beta}}{\sum_{c_{il} \in S_f} \tau_{il}^{\alpha} \eta_{il}^{\beta}} \qquad (12)$$

where S_f denotes the set of feasible edges immediately after the current partial path, τ_{ij} and η_{ij} are the pheromone and heuristic values respectively associated with edge c_{ij}, α and β are the parameters controlling the relative importance of pheromone versus heuristic information.

Further, when the ants completed their paths, the quality of their solutions is used to update the pheromone values of the edges. These updated values are then utilized by the ants in the next iteration to build new paths. This procedure continues until the maximum iteration number is reached or all ants tend to produce a similar path.

 ISBN: 978-1-61208-290-5

F. Extension to Finding Multiple Solutions

As is seen in this section, the optimization techniques with population-based transition are actually beam search, they can handle a set of feasible solutions at the same time. It follows that it would be relatively easy to modify or extend these techniques such that a set of optimal solutions rather than a single one will be returned from a single run of the algorithm. For example, the niching genetic algorithms were developed in [38] to find multiple interesting solutions in the job shop scheduling.

Finding multiple solutions as interesting trade-offs is also very important for multi-objective optimization tasks. Traditional ways to cope with multiple objectives is to build an overall objective function as weighted combination of individual objective values. However it is very hard to assign exact weights to reflect human preference, and therefore the final solution obtained may not be most preferred by human decision makers.

Multi-objective genetic algorithms have received intensive research for more than one decade (see [39][40], as examples). A nice feature of these algorithms is that a diversity of Pareto-optimal solutions can be found and presented to human decision makers, who then choose the most preferred alternative according to their preference. The Fast Non-dominated Sorting Genetic Algorithm (NSGA-II) [41] is an improved version of the early algorithms, which incorporates a fast non-dominated sorting algorithm and the crowding-distance assignment to improve the efficiency and effectiveness of the algorithm respectively.

More recently, multi-objective PSO algorithms were developed as extension of the single-objective counterparts. The key issue here is how to select global and local best particles in terms of multiple criteria. Different selection strategies have been proposed for this purpose. In [42] the tournament niche method was used to decide the global best particle, and the local best particle was identified according to Pareto-dominance. The other interesting strategy is to stochastically choose the global best particle from the non-dominated solutions using density-based probabilities [43].

V. CONCLUSION

This paper provides a survey of the principles and the state-of-the-art of numerical techniques for solving nonlinear optimization problems. All the techniques discussed are classified into two basic types: point-based transition and population-based transition, depending on whether a single point or multiple points are generated as new approximate solution(s) in each step of the iterations. Generally, the point-based approaches are simple and effective for optimization problems with a low number of parameters. However, when the dimension of the space increases, they become less efficient and are more likely to get stuck in a local optimum. In many practical applications, a point-based search method is often combined with the random sampling or multi-start strategy to increase the chance to find a global optimum. The population-based approaches are superior to the point-based ones in global search capability; they seem to be more suitable to be applied in high-dimensional search spaces. But, larger memory requirements and more computational cost are connected with them as side effects.

Most of the optimization approaches addressed here are derivative-free. This is a very useful property to promote wide applications in various situations without requiring the problem space to be continuous and differentiable. On the other side, some classical search methods such as gradient descent and Newton's method are also valuable and recommended to use, as long as the derivative information is available or achievable. The derivative-based methods are theoretically well founded and can contribute to substantial improvement of local search performance. The exploitation of derivative-based local search in a global evolutionary algorithm would be a promising direction of research for building new memetic computing frameworks.

Both the point-based and population-based approaches can be used to solve multi-objective optimization problems. For point-based approaches, it is necessary to construct an overall objective function as a weighted combination of individual objective values, and a single solution will be returned after the running of an algorithm. Since there is no clear relation between the weighting and the solution obtained, we cannot guarantee that the solution found is really the one that is most preferred by human decision makers. Comparatively, the population-based approaches appear to be more appropriate or have more potential for tackling problems with multiple objectives. As noted in Section IV, the population-based methods like GAs can easily be extended to deal with multiple objectives simultaneously and thereby returning a set of Pareto-optimal solutions rather than a single one. This enables human decision makers (designers) to choose the most preferred solution from a group of interesting trade-offs.

ACKNOWLEDGEMENT

The work is within the EMOPAC project granted by the Swedish Knowledge Foundation. We are also grateful to ABB FACTS, Prevas, and VG Power for co-financing the research.

REFERENCES

[1] L. Shi, S. Olafsson, and Q. Chen "An optimization framework for product design," Management Science, vol. 47, 2001, pp. 1681-1692 .

[2] M. S. Bazaraa, J. J. Jarvis, and H. D. Sherali, Linear Programming and Network Flows, 2nd Edtion, New York: John Wiley & Sons, 1990.

[3] D. G. Luenberger, Linear and Non-linear Programming. New York: Addison-Wesley, 1990.

[4] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty, Nonlinear Programming – Theory and Algorithms. New York: John

[5] L. G. Khachian, "A polynomial algorithm in linear programming," Soviet Mathematics Doklady, vol. 20, 1979, pp. 1093-1096Wiley & Sons, 1993.

 ISBN: 978-1-61208-290-5

[6] N. Karmarkar, "A new polynomial algorithm for linear programming," Combinatorica, vol. 4, 1984, pp. 373-395.
[7] W. H. Swann, "A survey of non-linear optimization techniques," FEBS Letters, vol. 2, 1969, pp. 39-55.
[8] A. H. G, Rinnoy Kan, G. G. E. Boender, and G. T. Timmer, "A stochastic approach to global optimization," In: K. Schnittkowski (Ed.) Computational mathematical programming, 1985, pp. 281-308.
[9] J. S. Arora, O. A. Elwakeil, and A. I. Chahande, "Global optimization method for engineering applications: a review,"Structural Optimization, vol. 9, 1995, pp. 137-159.
[10] S. Russel and P. Norvig, Artificial Intelligence: A Modern Approach (2nd ed.), New Jersey: Prentice Hall, pp. 111–114, 2003.
[11] N. Xiong and P. Funk, "Construction of fuzzy knowledge bases incorporating feature selection," Soft Computing, vol. 10, 2006, pp. 796 – 804.
[12] K. A. Sullivan and S. H. Jacobson, "Ordinal hill climbing algorithms for discrete manufacturing process design optimization problems," Discrete Event Dynamic Systems, vo1. 10, 2000, pp 307-324.
[13] M. Avriel, Nonlinear Programming: Analysis and Methods, Dover Publishing, 2003.
[14] R. Battiti, "First- and second-order methods for learning: Between steepest descent and Newton's method," Neural Computation, vol. 4, 1992, pp. 141-166.
[15] F. Glover, "Tabu search – Part one," ORSA Journal Computing, vol. 1, 1989, pp. 190-206.
[16] J. A. Bland and G. P. Dawson, "Tabu search and design optimization," Computer Aided Design, vol. 23, 1991, pp. 195-201.
[17] J. A. Bland, "Structural design optimization with reliability constraints using tabu search,"Engineering Optimization, vol. 30, 1998, pp. 55-74.
[18] J. M. Emmert, S. Lodha, and D. K. Bhatia, "On using tabu search for design automation of VLSI systems," Journal of Heuristics, vol. 9, 2003, pp. 75-90.
[19] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," Science, vol. 220, 1983, pp. 671-680.
[20] R. W. Eglese, "Simulated annealing: a tool for operational research.," European Journal of Operational Research, vol. 46, 1990, pp. 271-281.
[21] J. A. Nelder and R. A. Mead, "A simplex for function minimization," Computer Journal, vol. 7, 1965, pp. 308-313.
[22] D. E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, New York: Addison-Wesley, 1989.
[23] F. Herrera, M. Lozano, and J. L. Verdegay, "Tackling real-coded genetic algorithms: Operators and tools for behavioural analysis," Artificial Intelligence Review, vol. 12, 1998, pp. 265-319.
[24] F. Herrera, M. Lozano, and A. M. Sánchez, "A taxonomy for the crossover operator for real-coded genetic algorithms: An experimental study," International Journal of Intelligent Systems, vol. 18, 2003, pp. 309-338.
[25] N. Xiong, "Fuzzy rule-based similarity model enables learning from small case bases," Applied Soft Computing, vol. 13, 2013, pp. 2057-2064.
[26] N. Krasnogor and J. Smith, "A tutorial for competent memetic algorithms: model, taxonomy, and design issues," IEEE Trans. Evolutionary Computation, vol. 9, no. 5, 2005, pp. 474-488.
[27] Y. S. Ong and A. J. Keane, "Meta-Lamarckian in memetic algorithm," IEEE Trans. Evolutionary Computation, vol. 8, 2004, pp. 99-110.
[28] J. H. Ang, K. C. Tan, and A. A. Mamun, "An evolutionary memetic algorithm for rule extraction," Expert Systems with Applications, vol. 37, 2010, pp. 1302–1315.
[29] X. Zhang, A. Erdem, H. Shi, N. Xiong, D. Isovic, and M. Bobesic, "A novel memetic algorithm incorporating Nelder-Mead method in fuzzy controller design," Proc. Int. Conf. Computational Intelligence and Software Engineering, 2012, pp. 55-59.
[30] N. Xiong and P. Funk, "Combined feature selection and similarity modeling in case-based reasoning using hierarchical memetic algorithm," Proc. of the IEEE World Congress on Computational Intelligence, 2010, pp. 1537–1542.
[31] R. Storn and K. Price, "Differential evolution - A simple and efficient heuristic for global optimization over continuous spaces," Journal of Global Optimization, vol. 11, 1997, pp. 341–359.
[32] S. Das, "Differential evolution: A survey of the state-of-the-art," IEEE Trans. Evolutionary Computation, vol. 15, 2011, pp. 4-31.
[33] M. M. Ali, and A. Torn, "Population set based global optimization algorithms: Some modifications and numerical studies," Computers and Operations Research, vol. 31, 2004, pp. 1703 – 1725.
[34] J. Kennedy and R. Eberhart "Particle swarm optimization," Proc. IEEE Int. Conf. Neural Networks, 1995, pp. 1942–1948.
[35] M. Dorigo, V. Maniezzo, and A. Colorni, "The ant system: optimization by a colony of cooperating agents," IEEE Trans. Systems, Man, and Cybernetics – Part B, vol. 26, 1996, pp. 29-41.
[36] M. Dorigo, "Ant colony optimization," Scholarpedia, vol. 2, 2007, pp. 1461.
[37] M. Dorigo and L.M. Gambardella, "Ant Colony System : A cooperative learning approach to the traveling salesman problem," IEEE Trans. Evolutionary Computation, vol. 1, 1997, pp. 53-66.
[38] E. Perez, M. Posada, and F. Herrera, "Analysis of new niching genetic algorithms for finding multiple solutions in the job shop scheduling," Journal of Intelligent Manufacturing, 2012, vol. 23, pp. 341-256.
[39] C. M. Fonseca and P. J. Fleming, "Multiobjective optimization and multiple constraint handling with evolutionary algorithms, part I: A unified formulation," IEEE. Trans. Systems, Man, & Cybernetics, Part A, vol. 28, 1998, pp. 26-37.
[40] N. Srinivas and K. Deb, "Multiobjective function optimization using nondominated sorting genetic algorithm," Evolutionary Computation, vol. 2, 1995, pp. 221-248.
[41] K. Deb, A. Pratap, S. Agrawal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," IEEE Trans. Evolutionary Computation, vol. 6, 2002, pp. 182-197.
[42] D. S. Liu, K. C. Tan, C. K. Huang, C. K. Goh, and W. K. Ho, "On solving multiobjective bin packing problems using evolutionary particle swarm optimization," European Journal of Operational Research, vol. 190, 2008, pp. 357-382.
[43] P. K. Tripathi, S. Bandyopadhyay, and S. K. Pal, "Multi-objective particle swarm optimization with time variant inertia and acceleration coefficients," Information Sciences, vol. 177, 2007, pp. 5033-5049.

 ISBN: 978-1-61208-290-5

Improving the Performance of Particle Swarm Optimization Algorithm With a Dynamic Search Space

Benoît Vallade, Tomoharu Nakashima
Department of Computer Science and Intelligent Systems Osaka Prefecture University
Osaka, Japan
valladeben@cs.osakafu-u.ac.jp & tomoharu.nakashima@kis.osakafu-u.ac.jp

Abstract-**This paper addresses an improvement idea for Particle Swarm Optimization Algorithm (PSO). As a search algorithm, the PSO is used to tune a set of parameters and find the best combination of parameter values for this set. These parameters habitually take their values in a static search space. This paper proposes a solution to improve the efficiency of the algorithm with optimization problems using parameters, which take their values in dynamic space. The appreciable experiments' results prove that this one is an efficient solution to such problems.**

Keywords-algorithm of non-deterministic search; particle swarm optimization algorithm; dynamic search space

I. INTRODUCTION

Nowadays, in the aim to solve optimization problems, the algorithms of non-deterministic search are commonly used. These problems, such as robots' motion optimization [1], require to find the best combination of parameter values for the particular problem at hand. There are various kinds of search algorithms [2], such as tabu algorithms, genetic algorithms, PSO (Particle Swarm Optimization) algorithms, and others. Among all these algorithms, this paper will focus on the particle swarm optimization algorithm also called the PSO algorithm. This choice has been motivated by the high degree of adaptability of this algorithm which is the best choice to implement our improvement.

The concept of the PSO algorithm is based on the simulation of a simplified social model and more particularly on the animals flocking [3]. Its conception follows some standard which have evolved overtime [4].

Like the other algorithms of non-deterministic search, these standard PSO algorithms allow to tune a set of parameters, which take their values in static search space.

This means that any time during the optimization, the search space of each variable will stay the same.

However, some optimization problems use a set of variables, which take their values in dynamic search spaces [1]. This means that the search spaces of the variables may vary during the optimization.

This paper presents our solution to improve the efficiency of the PSO algorithms in case of problems using variables with dynamic search space. First, in the next section, we will describe the global concept of the search algorithms and detail the standard versions of the PSO algorithm. Next, the third section will explain in details the particularities of these dynamic problems and the algorithm's improvement used to solve them. The fourth section will give the results of some experiments which compare the efficiency of both algorithms, standard and new, on these problems. Finally, we will conclude on the quality of the PSO and the efficiency of the new algorithm.

II. STANDARD PARTICLE SWARM OPTIMIZATION

A. An algorithm of non deterministic search

As introduced before, the PSO algorithm is an algorithm of non-deterministic search. This means that it searches for the best combination of values for a set of variables. As the Table I shows, the variables take their values in search spaces defined by a minimal and a maximal values. These limits are given by the user and will take constant values.

TABLE I. SET OF VARIABLES' STATIC SEARCH SPACES

	Min	Max
A	-5	5
B	2	6
C	-10	-5
D	0	10
E	-2	5

In addition, it means that the algorithm follows the same global processes. Firstly, the algorithm generates a set of random solutions. A solution is a combination of values for the set of parameters. After that, the solution's quality will be determined through a fitness function. This function is completely dependent on the problem to be optimized and is given by the user. This quality value is used to compare the actual solution to the precedent best solution, and a new solution will be generated. These three steps (calculate solution's quality, compare solutions and generate a new solution) will be repeated so long as the optimization continues. This one stops when the stop criterion satisfies certain criteria chosen by the user (time, number of iterations, etc.). To finish, the generation of the new

 ISBN: 978-1-61208-290-5

solution depends on the algorithm (tabu search, genetic algorithm, PSO), but it generally uses the best and previous solutions.
This process is described in the Figure 1.

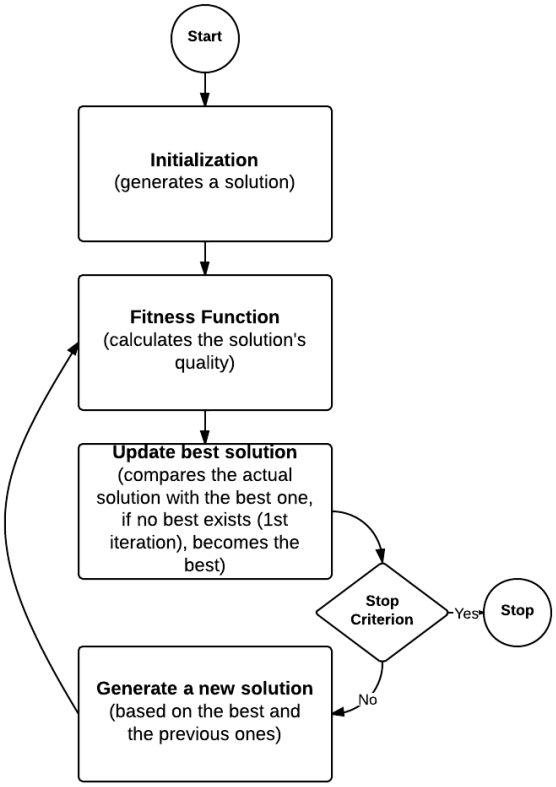


Figure 1. Global algorithm of an algorithm of non-deterministic search.

B. Concept overview

The special feature of the PSO algorithm is that its concept is based on the animals flocking [3].

As explained above, a solution is a combination of values for a set of parameters, while a problem is defined by these parameters, which take their values in static search spaces. Consequently, another way to represent an optimization problem is to consider a solution as a position in a search space defined by crossing all the individual parameters' search space. The position takes place in a hyper-space of dimension equals to the numbers of parameters. And its coordinate's values correspond to the parameter's value of the solution.

Coming back to the animal's social behaviour subject and more particularly on the birds flocking; during their feeding time, multiple birds evolved in the same space and search for the position where there is the biggest quantity of foods. In the course of their search, each bird always remembers the position where they have found the biggest quantity of food. In addition, as the birds follow a social behaviour inside the flock, they also share the best position found by the whole flock. Finally, as shown in Figure 2, each bird adapts its movements in the search space according to these knowledge.

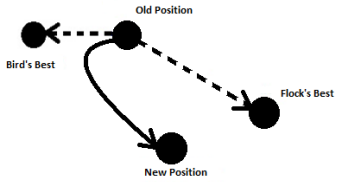


Figure 2. Birds' movements according to Best Positions knowledge.

This social behaviour is used by the PSO algorithm as a conceptual idea to generate new solutions and optimize the set of parameters. In this transposition, the birds will be called particles, the flock will be the swarm and the quantity of foods will correspond to the quality of the solution. As the flock of birds seeks for the best food's position, the swarm of particles seeks for the best quality's position.

C. Standard algorithm

Our research is based on 2011's version of the standard PSO algorithm described in the paper of Maurice Clerc [4], with the particularity of not using the neighbourhood system (in case of neighbourhood system, the particles are grouped in teams and they share the information about the best position found by all the team's member only inside the team, in our case there is only one big team, which correspond to the whole swarm). This part of the paper gives some details about this version of standard PSO algorithm.

1) Particle's components and algorithm: As explained in the previous part on the birds flocking transposition, a swarm of particles is included in the search space. Each particle is aware of:

- Its Position (initialized randomly in the search space)
- Its Velocity (initialized randomly in the search space)
- Its Best Position ever found (initialized as the first particle's position)
- The Swarm's Best Position (initialized by comparing all the quality Particle's first position)

It should be noted that in the 2011 version, the initialization of the positions' and velocities' values are randomly generated, parameter by parameter.

Each iteration of the optimization, these particles' attributes are updated following the process described in Figure 3:

 ISBN: 978-1-61208-290-5

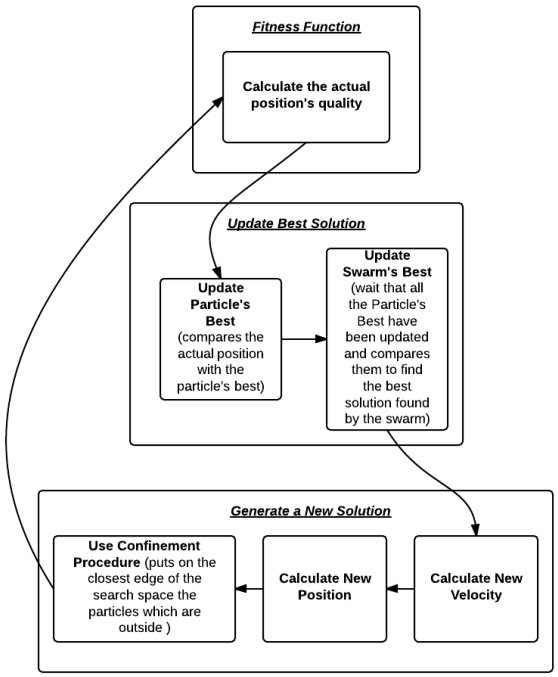


Figure 3. Process of an iteration of the PSO algorithm.

2) Evolution rules: Below are given the equations used for the velocity and position update and all the other details useful to the implementation of the 2011 version of the PSO algorithm.

a) *Swarm size and initialization:* In the 2011 version the swarm size (number of particles) will be user defined, with 40 as suggested value. The initialization of the particles' attributes will be conducted as described before.

b) *Calculate Velocity and Position:* First the velocity will be updated by using the following equations:

$$G_i = x_i + c * \left(\frac{p_i + l_i - 2x_i}{3}\right) \quad (1)$$

$$H_i(G_i, \|G_i - x_i\|) \quad (2)$$

$$v_i(t+1) = wv_i(t) + x'_i(t) - x_i(t) \quad (3)$$

In these equations:

- c = 1.193 and w=0.721 (constants)
- x_i (or $x_i(t)$) is the actual position value of the i^{th}particle
- p_i is the particle's best position of the i^{th}particle
- l_i is the swarm's best position of the i^{th}particle
- G_i is the centre of gravity of the three points x_i, p_i and l_i.
- H_i is the hyper sphere of centre G_i and radius $G_i - x_i$
- $x'_i(t)$ is a position randomly choose in the hyper sphere H_i
- $v_i(t)$ is the actual velocity of the i^{th}particle
- $v_i(t+1)$ is the new velocity of the i^{th}particle

Next, the position is updated by using the equation:

$$x_i(t+1) = wv_i(t) + x'_i(t) \quad (4)$$

In cases where $l_i = p_i$, the following gravity centre equation is used for the velocity updating:

$$G_i = x_i + c * \left(\frac{p_i - x_i}{2}\right) \quad (5)$$

c) Confinement: But, sometimes, the new position of the particle is out of the search space. In those cases, the algorithm uses a confinement procedure, which moves the particle on the closest edge of the search space. This movement is conducted by replacing the value of each parameter of the position by the closest corresponding parameter's search space limits, min or max. Finally, the velocity forced to the following value:

$$v_i(t+1) = -0.5 * v_i(t+1) \quad (6)$$

d) *Particle's and Swarm's Best:* To finish the quality of the new position is calculated by using the fitness function. As said before, this function depends on the problem and is defined by the user. Its results will be used to compare the different positions found by the algorithm. During a first comparison the value of the particle's best position is updated in function of the previous one and of the actual position. In order to do the second comparison, the algorithm waits that all the particles' best of the swarm are updated. All the particles' best position of the swarm will be compared to determine which the swarm's best position is, and this knowledge will be shared with all the swarm's particles.

III. PSO IN DYNAMIC SEARCH SPACES

This section discusses the topic of the problems based on set of parameters with dynamic search spaces. And then the proposed solutions to deal with such problems and through the changes made on the standard PSO algorithm are explained.

A. *Dynamic Search Space problems*

Contrarily to the previous standard types of problems, which used parameters taking their values in static search spaces, some problems are based on dynamic search spaces. In these kinds of problems, the search spaces limit of some parameters depends on the value of other parameters. Table II gives an example of such a set of parameters:

 ISBN: 978-1-61208-290-5

TABLE II. SET OF VARIABLES' DYNAMIC SEARCH SPACES

	Min	Max
A	0	5
B	-A	A
C	-5*A+B	+5*A+B
D	-15	2*A
E	-20	10

In those kinds of problems, commonly used in robots' motion optimization [1], the search spaces limits depends on the value of the parameters and consequently on the position of the particle. Finally, as shown in Figure 4, all the particles evolved in a different search space, which changes in function of the particle's position. However, a maximal space search can be created by using the maximum value possible for each parameter's maximum limit and the minimum value possible for the parameter's minimum limit.

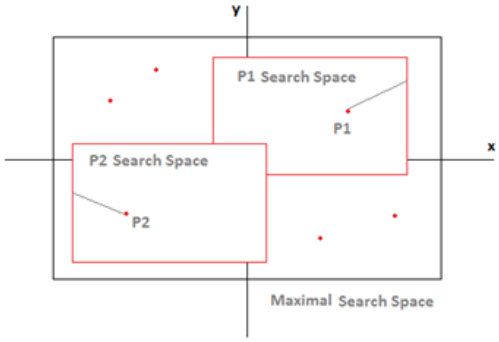


Figure 4. Dynamic search space representation in a two-parameter optimization problem.

Table III uses the Table II 's example to create the corresponding maximal search space.

TABLE III. SET OF VARIABLES' MAXIMAL SEARCH SPACES

	Min	Max
A	0	5
B	-5	5
C	-30	30
D	-15	10
E	-20	10

This space consequently contains all the individual search spaces possible. By individual search space, we mean the search spaces which are created by using the position value. But it also contains positions which are not included in these individual search spaces. To keep the precedent example (Table II and Table III), the position of coordinate (0;-5;0;0;2) is available in the maximal search space but not in individual spaces. As the search spaces limits are user-defined, we will call these positions, "uninteresting positions".

B. *Dynamic search space PSO concept*

To solve such problems, there are two options. The first one is to apply the optimization on the maximal search space. As described above, this space includes all the search spaces possible and has the particularity to be static. The advantage of such a solution is that, as the search space is static, the standard PSO algorithm described before can be used. The disadvantage is that the optimization will also be conducted on uninteresting positions. This may result in loss of time and a final optimization position not intended by the user.

The second solution is to use individual search spaces. This means that each particle will have its own search space and this one will change as the same time as the particle move. This solution avoids the search on the uninteresting positions but implies some modifications to the standard algorithm.

As we chose to improve the efficiency of the PSO algorithm in case of dynamic search space problems, we will explain in the following part the necessary changes to the standard algorithm.

C. *Dynamic search space PSO modifications*

This part discusses about the problem faced by the standard algorithm resulting from the choice of the second solution and about the possible modifications to avoid it.

1) *Problem:* During the initialization step as well than during the confinement methods, the new value of the position will be generated parameter by parameter. A random value will be generated between the parameter's search space limits for the initialization and the closest limit will be searched for the confinement. But, by using dynamic search spaces, these limits values will depend on others parameters' values. Also, the algorithm would not be able to generate a parameter's value if its limits have not ever been defined. That is why the parameters' values need to be defined in the good order.

2) *Modification:* This order will of course be based on the links between the parameters. The parameter A is linked with the parameter B if value of B is required to calculate the limits of A's search space.

In the aim to represent these links, we chose to use an acyclic graph representation.

These graphs are tree graphs with the particularity to allow multiple roots and multiple parents for a same child. Of course we can't allow cycle due to the impossibility to generate a position value if the parameters are linked through a cycle. In this case, the first proposed methods using the maximum search space should be used.

Our implementation uses a unique root, which does not correspond to any parameter, but it allows us to insert all the parameters in the same graph, even the one which are not linked with others parameters (search space limits have

 ISBN: 978-1-61208-290-5

constant value). These special parameters will be direct leaf of the root.

Each parameter is represented by a node which is linked to other nodes following the parameters links. The node A is parent of the node B if the parameter A needs the value of B. The nodes will be sorted in depth layers, a node will also be in the layer under the layer of its deepest parents. The direct leafs are inserted in the deepest layer of the graph.

Finally, to generate the position value, the order to follow is decided by using the graph. This path corresponds to a back breadth-first search of the graph. This means that we start from leafs and we head to the root of the graph by visiting each node of a layer before to go to the upper one. node includes calculate its search space limits and generate its value (or search the closest limits for the confinement method). Figure 5 is the acyclic graph representation of the example given in Table II. It also shows the calculation path (dotted arrows):

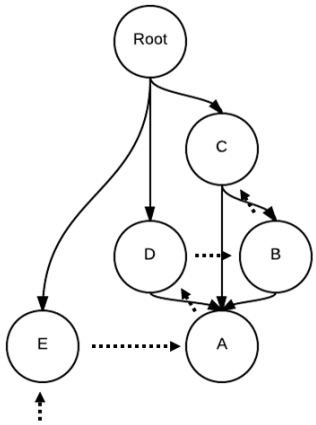


Figure 5. Acyclic graph representation of parameters link and calculation path.

IV. EXPERIMENTS

In order to compare the efficiency between the standard and the new algorithm on dynamic search spaces problems, we set up two experiments. The problem is that there are no such problems in literature [5]. Consequently, we cannot compare it with the other algorithms' results but have to create our own artificial problems. This is the why the two next experiments do not correspond to any known problems and have no real correspondence with the real life. These problems have only been designed to proof the functionality and efficiency of the new algorithm compared with the standard algorithm.

A. Experiment 1

For this one, we will use a swarm of 40 particles and launch optimization of 500 iterations. We created three optimization problems very simple (so they will be solved in 500 iterations) and compared how many iterations are required to find the optimal position (position with the best fitness quality).

These problems have the following characteristics:

- two dimensional problems (parameters: X , Y)
- The Y's search space limits depends on X's value
- The optimal position is the point of coordinate (5.5 ; 0.01) and the fitness function calculate the distance between the particle position and the optimal position.
- X's value vary in [0 ; 1000]
- Y's vary in the limits defined by Table IV, the standard algorithm will use the maximum search space:

TABLE IV. SEARCH SPACE LIMITS FOR THE Y PARAMETER

	New Algorithm		Standard Algorithm	
	Min	Max	Min	Max
A	$-(0.05 * x) + 0.25$	$+(0.05 * x) - 0.25$	-49.75	49.75
B	$-(0.25 * x) + 1.25$	$+(0.25 * x) - 1.25$	-240.5	240.5
C	$-(0.5 * x) + 2.5$	$+(0.5 * x) - 2.5$	-497.5	497.5

We repeated the optimization 10 times and took the average number of iterations needed to find the optimal point (as we calculate the distance, the fitness value = 0). Table V regroups the results of this experiment:

TABLE V. AVERAGE NUMBER OF ITERATIONS NEEDED TO FIND THE OPTIMAL POSITION

	A	B	C
Standard algorithm	279	297	298
New algorithm	282	255	250

We remark that the standard algorithm has better results for the problem A, but becomes less efficient on B and C. This means that the new algorithm would be more efficient on big search spaces, and more particularly, when the number of uninteresting positions grows up, which make sense.

B. Experiment 2

As the previous results seems indicate that the new algorithm is more efficient on big search space we set up a second experiment to confirm. To do so we still used the same configuration for the PSO algorithm (40 particles, 500 iterations and we created a more complicated problem evolving on a bigger maximal search space. The search spaces limits of this problem are described in Table VI and the parameters links can be visualized in Figure 6.

 ISBN: 978-1-61208-290-5

TABLE VI. SEARCH SPACE LIMITS FOR THE SET OF PARAMETERS

Parameters	Min	Max
A	-(D+E+F)	D+E+F
B	-(E+F+G)	E+F+G
C	-(H+I)	H+I
D	-500	500
E	-2.5*J	2.5*J
F	-0.5*J	0.5*J
G / H	-500	500
I	-(K*0.5)	K*0.5
J / K / L	-500	500

It should be noted that the maximal search space is not given, but it can be easily calculated, as shown in the previous examples.

Also to be noted that the optimal position is a static position (0,0...,0), centre of the search space. The fitness function calculates the distance to this point, so the best quality value possible is 0.

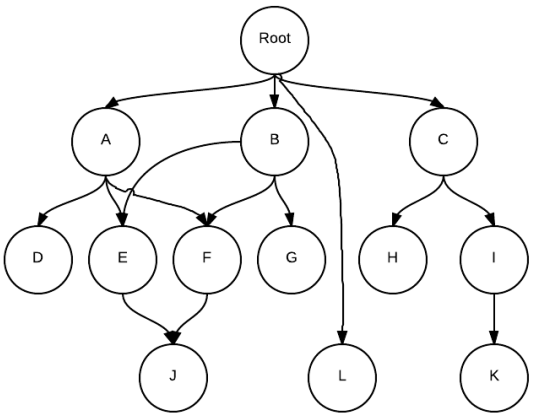


Figure 6. Acyclic graph representation of parameters link.

In this experiment, the problem is too big to be solved in 500 iterations; so, we compare the quality of the solutions. The standard algorithm gave an average of 0.0027, while the new algorithm gave an average of $7.90 * 10^{-25}$. The results of this experiment show clearly the efficiency of the new algorithm.

V. CONCLUSION AND FUTURE WORK

To conclude, the PSO is a very simple and easily adaptable algorithm. The actual standard version of the PSO algorithm is able to deal with dynamics search space by venturing a loss of time and falling on an uninteresting result. This paper described an efficient solution to improve its performance in this case. However, there is actually no efficient solution for dynamic search space problems, where parameters are cycled linked. Our future objectives will be to be able to deal with cycled graph, and to test our solution on real world problems.

REFERENCES

[1] T. Uchitane and T. Hatanaka, “Applying evolution strategies for biped locomotion learning in roboCup 3D soccer simulation”, Proc. of 2011 IEEE Congress on Evolutionary Computation New Orleans. LA, 2011, pp. 179-185.

[2] H. Youssef S. M. Sait, and H. Adiche, “Evolutionary algorithms, simulated annealing and tabu search: a comparative study”, Engineering Applications or Artificial Intelligence, 2001, pp. 167-181.

[3] J. Kennedy and R. Eberhart, “Particle swarm optimization”, Proc. of IEEE International Conference on Neural Networks Perth, 1995, pp. 1942-1948.

[4] M. Clerc, “Standard particle swarm optimisation”, technical report, 2012.

[5] M. Molga and C. Smutnicki, “Test functions for optimization needs”, 2005.

 ISBN: 978-1-61208-290-5

Reliable Outer Bounds for the Dual Simplex Algorithm with Interval Right-hand Side

Christoph Fünfzig
Fraunhofer Institute for Industrial Mathematics
Kaiserslautern, Germany
Email: c.fuenfzig@gmx.de

Dominique Michelucci, Sebti Foufou
Le2i, University of Burgundy
Dijon, France
Email: {dominique.michelucci, sebti.foufou}@u-bourgogne.fr

***Abstract*—In this article, we describe the reliable computation of outer bounds for linear programming problems occuring in linear relaxations derived from the Bernstein polynomials. The computation uses interval arithmetic for the Gauss-Jordan pivot steps on a simplex tableau. The resulting errors are stored as interval right hand sides. Additionally, we show how to generate a start basis for the linear programs of this type. We give details of the implementation using OpenMP and comment on numerical experiments.**

***Keywords*-verified simplex algorithm; interval arithmetic; tableau form; OpenMP parallelization**

I. INTRODUCTION

Linear relaxation [1] is a common method to solve non-linear systems in several variables with domains $D_i \subset \mathbb{R}$, $i = 1, \ldots, N$. For the system with variables $x_1 \in [0,1]$ and $x_2 \in [0,1]$,

$$x_1^2 - x_2 = 0$$
$$x_2 - x_1 \leq 0$$

a linear relaxation derived from the tangent plane in $x_1 = x_2 = 0.5$ is

$$(x_1 - 0.5) - (x_2 - 0.5) - 0.25 \geq 0$$
$$x_2 - x_1 \leq 0$$

In [2], we presented linear relaxations for the monomials x_i^2 and $x_i x_j$, $i < j$ with $x_i \in D_i$, $x_j \in D_j$, which are derived from the Bernstein polynomials on domain D. The curves (x_i, x_i^2), and the surfaces $(x_i, x_j, x_i x_j)$, $i < j$ are enclosed in a polytope, called *Bernstein polytope* (Figure 2).

With linear relaxation, a quadratic system

$$F(x) = 0,\ G(x) \geq 0,\ x = (x_1, \ldots, x_N)$$

gives a linear system in the variables x_i, x_{ii}, and x_{ij}, $i < j$. For example, a lower bound for component i can be obtained by solving a linear program: minimize x_i on the linear system obtained with the Bernstein polytope for domain D.

In this article, we show how to use interval arithmetic in a tableau-form implementation of the dual simplex algorithm (Section II) to verify computations and to generate a tight lower bound on the minimum value. The tableau has floating-point entries and uses an interval right-hand side. In a pivot operation of the Gauss-Jordan algorithm, rounding errors are collected and stored in the right-hand side intervals (Section II-B). For the application of linear relaxations using the Bernstein polytope, we give two ways to generate a start basis for the occurring linear programs in Section II-C. Finally, we conclude on this work in Section IV.

A. Related Work

In an overview, there are three classes of methods for solving polynomial systems. Using computer algebra, polynomial expressions (*resultants*) can be defined, which are equal to zero if and only if the polynomials have a common root. They provide simple and effective methods for low degree problems due to the degree of the resulting expression. Decomposing an arbitrary polynomial in the ideal of the given polynomials is possible with a special generator of the ideal. Such an ordered generator (*Gröbner basis*) can be used to compute common polynomial factors. From a computational point of view, these methods need exact arithmetic. Recently, some articles [3] work towards their numerical computation with tools from interval arithmetic.

An established semi-numerical method solves a given polynomial system P with the same number of equations as variables. The method uses a continuation method [4] with a scalar parameter $t \in [0,1]$ to deform a polynomial system P^0 with known solutions to the given system P. Finding an initial simple polynomial system P^0 for the given P is the main difficulty of the method.

In this article, we solve quadratic polynomial systems by *branch-and-bound* using a linear relaxation for the monomials. We compute an outer bound for the optimum value of the linear program reliably. In [2], the revised simplex code *SoPlex* [5] in floating point arithmetic and a backward analysis of the final linear system for the objective value was used. The article [6] gives a comparison of linear programming codes using rational arithmetic and floating-point arithmetic. Of course, the use of floating-point arithmetic, which is used at least in parts of the code, is significantly faster than exact rational arithmetic. [7] describes how to compute a lower bound using an arbitrary linear program solver. The authors use the *weak optimality theorem of linear programming*: Any feasible point y of the dual problem $A^t y \leq c$ ($\max b^t y$) gives a lower bound $b^t y$ for the minimum of the primal problem $Ax = b$, $x \geq 0$ ($\min c^t x$). As outlined in [7], the lower bounds obtained from a verified, feasible point can be away from the optimum value for ill-conditioned problems. The article [7] additionally

 ISBN: 978-1-61208-290-5

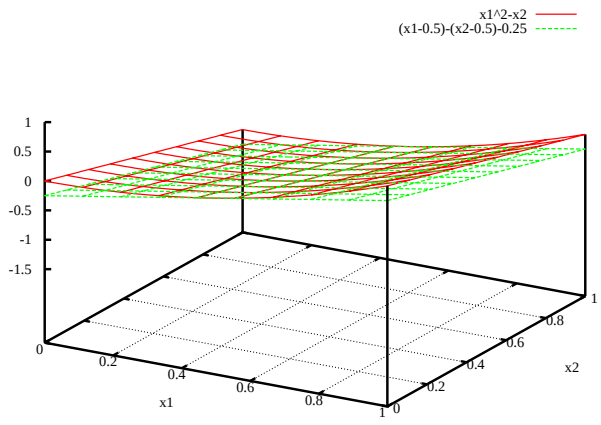


Fig. 1: Linear relaxation (dashed below) for $x_1^2 - x_2$ (solid above).

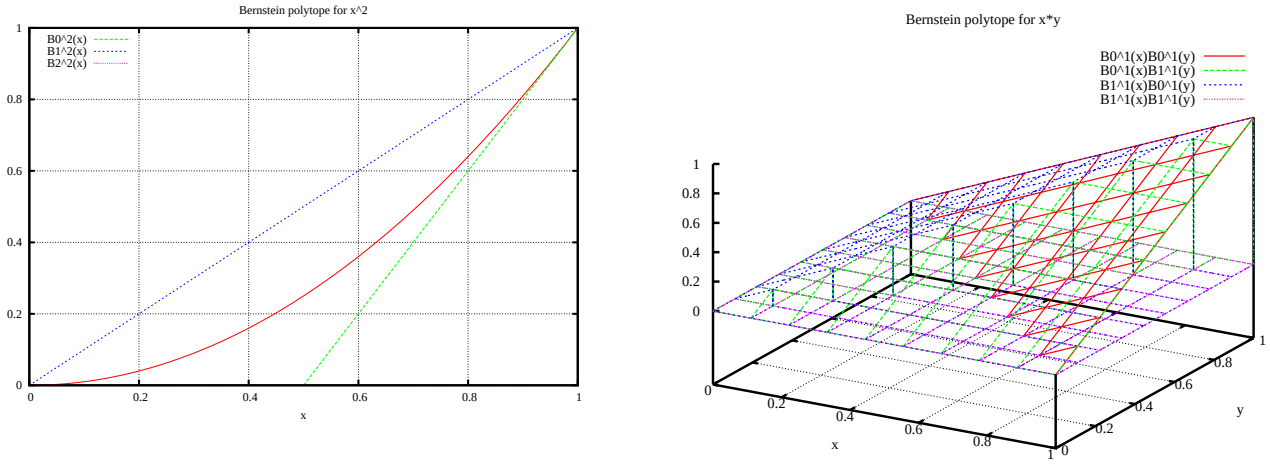


Fig. 2: Bernstein polytope enclosing the curve (left) (x, x^2): $B_0^2(x) \geq 0$, $B_1^2(x) \geq 0$, $B_2^2(x) \geq 0$. The surface (right) (x, y, xy): $B_0^1(x)B_0^1(y) \geq 0$, $B_1^1(x)B_0^1(y) \geq 0$, $B_0^1(x)B_1^1(y) \geq 0$, $B_1^1(x)B_1^1(y) \geq 0$.

considers the computation of upper bounds. Concerning the parallelization of simplex codes, [8] summarizes a number of attempts. The authors of [9] consider the parallelization of the sparse dual simplex code *CPLEX*. They make out the steepest-edge pricing rule as a good candidate for parallelization, which compares all infeasible rows based on the resulting change of the dual variables.

B. Notation

We use $\underline{R}$ for the lower bound of an interval R and $\overline{R}$ for the upper bound. median(R) denotes the median of the interval bounds $\underline{R}$ and $\overline{R}$ computed as $[0.5\,(\underline{R} + \overline{R})]^-$. For a real number a, we denote by a^- the largest floating-point number smaller or equal to a, and by a^+ the smallest floating-point number larger or equal to a. We denote by e_k the kth vector of the canonical basis with $e_{k,k} := 1$, and 0 otherwise. As usual, an inequality between vectors, like $x \geq 0$, applies to all components i: $x_i \geq 0$.

II. Linear Programming Problem

A linear program in standard form is defined by

$$\begin{aligned} &\min c^t x \\ &Ax = b \\ &x \geq 0 \end{aligned}$$

where A is a $m \times n$ real matrix, b is a m-component real vector, and c is a n-component real vector. The system $Ax = b$ contains the linear equality constraints, and the function $c^t x$ defines the linear objective function to be minimized. An inequality $a_1^t x \leq b_1$ is transformed into an equality by a new variable $x_s \geq 0$: $a_1^t x + x_s = b_1$, which is called a *slack variable* [10]. Note that in our case it is $m \leq n$, i.e.,

our problem has at least as many variables as rows due to the slack variables. There are several ways to ensure non-negativity of variables. The first way is to use symbolic substitution of the problem variables $\tilde{x_i} \rightarrow (x_i - \underline{D_i})$ if $\underline{D_i} < 0$ and change all equations and inequalities into problem variables $\tilde{x_i} \in [0, \overline{D_i} - \underline{D_i}]$. This adds a linear number of additional terms to the given problem. The second and preferred way is to substitute all variables x_i and x_{ij} locally into $\tilde{x_i} \rightarrow (x_i - \underline{D_i})$ resp. $\tilde{x_{ij}} \rightarrow (x_{ij} - \underline{D_i \cdot D_j})$ so that $x_i = \tilde{x_i} + \underline{D_i}$ resp. $x_{ij} = \tilde{x_{ij}} + \underline{D_i \cdot D_j}$). Then row r changes into $\sum_k a_{rk}(\tilde{x_k} + \underline{D_k}) = b_r$ which needs to be done only once during tableau setup.

The tableau-form implementation of the simplex algorithm (with column basis) selects a maximal subset of m linear independent columns of A (corresponding to the basic variables of x), where m is the rank of the matrix A. The subset with index set B is called a *basis*, and the corresponding submatrix $A_{*,B}$ is invertible; the rest matrix is denoted by $A_{*,N}$. Non-basic variables always have a zero value. A basis update operation maintains the reduced row-echelon form of the tableau ($x_B = A_{*,B}^{-1} b$)

$$\begin{pmatrix} c_B^t x_B & (c_B - c_B^t A_{*,B}^{-1} A_{*,B})^t & (c_N - c_B^t A_{*,B}^{-1} A_{*,N})^t \\ x_B & A_{*,B}^{-1} A_{*,B} & A_{*,B}^{-1} A_{*,N} \end{pmatrix} \quad (1)$$

which allows to look up the reduced costs in the first row, the objective function value in the first column of the first row, and the basic variable values x_B in the first column below the first row of this tableau [10]. Furthermore, we group matrix rows into equalities (without a slack variable) given by the index set E and inequalities (each having a slack variable) given by the index set I. Maintaining this form is possible with the Gauss-Jordan algorithm from numerical linear algebra. As tableau rows define basis variables, it is possible to check the non-negativity constraints $x \geq 0$ and to select a leaving variable in the simplex algorithm (*pricing* [10]). The leaving variable is replaced by an entering variable, which can be selected from the reduced costs in the first tableau row (*ratio test* [10]). For applications, this *dual form of the simplex algorithm* is beneficial [11], which changes an infeasible basis with a sub-minimum value into an optimum, feasible basis. It selects the leaving variable from the infeasible basis first and replaces it by the entering variable. In contrast the *primal form of the simplex algorithm*, selects the entering variable first, then selects the leaving variable until an optimum value is reached.

A. Pricing Rule and Ratio Test

Important for the performance of the solver is the pricing rule, the ratio test, and the start basis [11]. For the pricing rule, we consider the *steepest-edge* rule

Definition 1 (Goldfarb-Forrest pricing rule): Select row r which has the most negative ratio $\frac{x_r}{|e_r^t A_{*,B}^{-1}|_2}$.

where $d = e_r^t A_{*,B}^{-1} A_{*,N}$ is the change of the dual variables per unit change of x_r (see Equation 1). The values x_r are stored as right-hand sides described in the following Section II-B, and A_{*B}^{-1} is part of the tableau.

For the ratio test, we use

Definition 2 (Harris ratio test): Select column s so that $a_{r,s}$ is minimum with $\frac{c_j}{a_{r,j}} \geq \theta_r(\epsilon)$, $a_{r,j} < 0$, $\theta_r(\epsilon) := \min\{\frac{c_j+\epsilon}{a_{r,j}} : a_{r,j} < 0\}$.

This rule chooses the element $\frac{c_s}{a_{r,s}}$ of the set $\{\frac{c_j}{a_{r,j}} \geq \theta_r(\epsilon) : a_{r,j} < 0\}$ defined by a small parameter $\epsilon > 0$. This allows to choose the denominator $a_{r,s} < 0$ with largest magnitude for the division. Note that c_j is non-negative up to rounding errors for the dual simplex algorithm always. The ratio test traverses the objective function row and the row r of the tableau in parallel.

B. Pivot Steps using Interval Arithmetic

For the collection of computer arithmetic errors during a pivot step of the Gauss-Jordan algorithm, we use interval arithmetic. Let $a_{r,s}$ be the pivot element in row r, and D_i the variable domain for variable x_i. Then the linear equation

$$\sum_j \frac{a_{r,j}}{a_{r,s}} x_j = \frac{b_r}{a_{r,s}}$$

transforms into an interval equation

$$\sum_j R_{r,j} x_j = R_r$$

where we can select a floating-point value $a_{r,j} \in R_{r,j}$ of the interval so that $R_{r,j} \subset a_{r,j} + [(\underline{R_{r,j}} - a_{r,j})^-, (\overline{R_{r,j}} - a_{r,j})^+]$. The intervals can be collected and stored as an interval right hand side R'_r

$$\sum_j a_{r,j} x_j = R_r - \sum_j [(\underline{R_{r,j}} - a_{r,j})^-, (\overline{R_{r,j}} - a_{r,j})^+] D_j =: R'_r \quad (2)$$

With the representative $a_{r,j} := \text{median}(R_{r,j})$, the resulting interval $[(\underline{R_{r,j}} - a_{r,j})^-, (\overline{R_{r,j}} - a_{r,j})^+]$ has smallest width. Figure 3 shows the hyperplane arrangement for an example.

Similarly, a row operation as required in the Gauss-Jordan algorithm between row r and row i

$$\sum_j (a_{i,j} - a_{r,j} \frac{a_{i,s}}{a_{r,s}}) x_j = b_i - b_r \frac{a_{i,s}}{a_{r,s}}$$

can be performed in interval arithmetic

$$\sum_j R_{i,j} x_j = R_i$$

and rewritten using an interval right-hand side

$$\sum_j a_{i,j} x_j = R_i - \sum_j [(\underline{R_{i,j}} - a_{i,j})^-, (\overline{R_{i,j}} - a_{i,j})^+] D_j =: R'_i \quad (3)$$

In this form, a sufficient condition for the feasibility of x_i, $i \in B$ is $\underline{R_i} \geq \underline{D_i}$. In case $\overline{R_i} < \underline{R_i}$, it is infeasible and a candidate for the pricing rule. Otherwise some R_i have smaller and larger than $\underline{D_i}$, in which case we stop the solving process with a lower bound of the optimum value. Due to use of interval right-hand sides, it is possible to reduce the number of variables $\{x_j\}$ by replacing a variable x_j with its interval

 ISBN: 978-1-61208-290-5

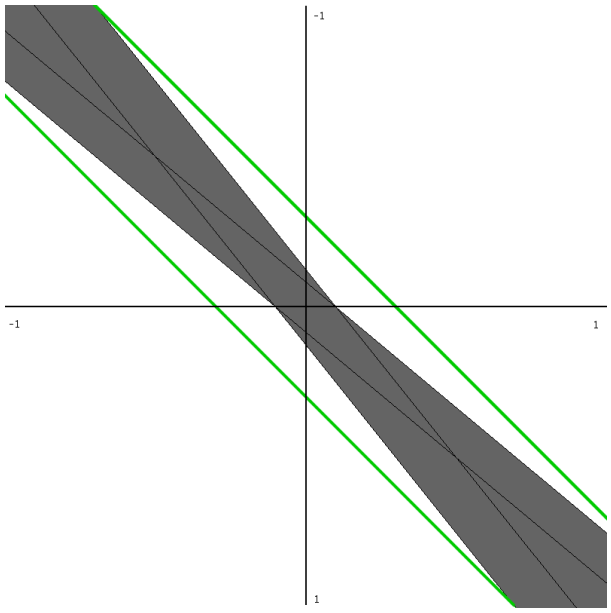


Fig. 3: Hyperplane arrangement (grey) for the interval equation $[0.8, 1.2]x + [1.0, 1.0]y + [-0.5, 0.5] = 0$. A thick hyperplane results from the selection of interval representatives $1.0x + 1.0y + [-0.7, 0.7] = 0$.

D_j. In this way, an active subset of variables of the linear program can be defined.

In the geometric view of the polytope defined by interval right-hand sides, the thick hyperplanes bound a set of polytopes, which are not necessarily of the same topology. See Figure 4 for an example, where the minimum-y vertex of the outer hyperplanes is defined by the intersection of r_1 and r_3, but the minimum-y vertex of the inner hyperplanes is defined by the intersection of r_1 and r_2.

Note that these topology changes make situations possible, where the outer polytope is non-empty but the inner polytope is empty. In such cases, the algorithm can not decide feasibility of the linear program.

C. Start Basis Generation

Inside the non-linear solver, we only have to handle objective functions of the form $x_i = e_i^t x$ for a variable index i. Start basis generation can be done by performing the Gauss-Jordan algorithm on the equation part $A_{E,*}$ of the system. This defines a subset B_E of the basis B. Note that the set B_E depends on the pivot selection strategy in the Gauss-Jordan algorithm and does not need to change during pivoting in the simplex algorithm. Also pricing rule and ratio test only consider the inequalities I and the inequalities of the Bernstein polytope.

The first possibility is to select only pivots with column index different from i. In this case, where B_E does not contain variable index i, we can easily complete the basis from the vertex with smallest value x_i of the Bernstein polytope for (x_i, x_i^2). I.e., for minimization (with $B_i^{(2)}$ the i-th quadratic Bernstein polynomial)

$$\begin{aligned}
(B_0^{(2)}(x_i) &\geq 0)\\
B_1^{(2)}(x_i) = 2(-x_{ii} + \boxed{(u_i + v_i)x_i} - u_i v_i) &\geq 0\\
B_2^{(2)}(x_i) = \boxed{x_{ii}} - 2u_i x_i + u_i^2 &\geq 0
\end{aligned}$$

for maximization

$$\begin{aligned}
B_0^{(2)}(x_i) = x_{ii} + \boxed{-2v_i x_i} + v_i^2 &\geq 0\\
B_1^{(2)}(x_i) = 2(\boxed{-x_{ii}} + (u_i + v_i)x_i - u_i v_i) &\geq 0\\
(B_2^{(2)}(x_i) &\geq 0)
\end{aligned}$$

The second possibility is to select a pivot with column index i. In this case, where B_E contains the variable index i, we can generate a start basis from the equation row k defining variable x_i. Let $x_i + \sum_{j \neq i} a_{k,j} x_j = R_k$ be row k. If there are columns $a_{k,j} > 0$ the current basis part B_E is not optimum, and it can be changed by primal steps into an optimum basis. I.e., for each such column j with $a_{k,j} > 0$ we determine a row r such that $a_{r,0} \geq 0$ and $-a_{r,0}\frac{a_{k,j}}{a_{r,j}} < 0$ is minimum. Both strategies are compared in Section III based on a numerical example.

objective function	
user equalities, n vars	$u + 3c + 4d$ slack vars
u user inequalities, n vars	$u + 3c + 4d$ slack vars
$3c$ Bernstein inequalities	$u + 3c + 4d$ slack vars
$4d$ Bernstein inequalities	$u + 3c + 4d$ slack vars

Fig. 5: Tableau organization with regions for the user system and for the Bernstein polytope.

III. Implementation and Numerical Experiments

We implemented the dual simplex algorithm in C/C++ using the tableau organization shown in Figure 5. The tableau can be stored as an $m \times n$ array of `double`-entries or in a sparse form as an array of m rows of index/entry pairs. The right-hand side vector is represented using the *boost* interval arithmetic library. We additionally keep a basis description consisting of arrays, *var* giving the index of the defining row for a variable index, and *row* giving the index of the variable defined in a row. Both are inverse to each other: $row[var[j]] = j$ for variable index j and $var[row[i]] = i$ for row index i. We have one pivot operation $\frac{1}{a_{r,s}} A_{r,*}$ for the pivot row r, and m row operations $A_{i,*} + f_i A_{r,*}$ for all other rows $i \neq r$. The m different row operations can be done in parallel using OpenMP. We use the basis description to exclude basis columns, which are unit vectors and therefore contain a zero value in the pivot row r. Altogether, the number of multiplications and additions for a row operation ranges from $n-m$ to n. Note that in the interval version (Equations 2 and 3), the pivoting and row operations are not entrywise as in [12] but require a reduction operation for the right-hand side interval. We avoid a parallel reduction by storing the right-hand side updates $[(\underline{R_{i,j}} - a_{i,j})^-, (\overline{R_{i,j}} - a_{i,j})^+]D_j$ in an array and perform the summation sequentially. Similarly, we perform the loop for the steepest edge pricing rule and the loop for the ratio test in parallel using OpenMP but without a critical section for the minimization.

 ISBN: 978-1-61208-290-5

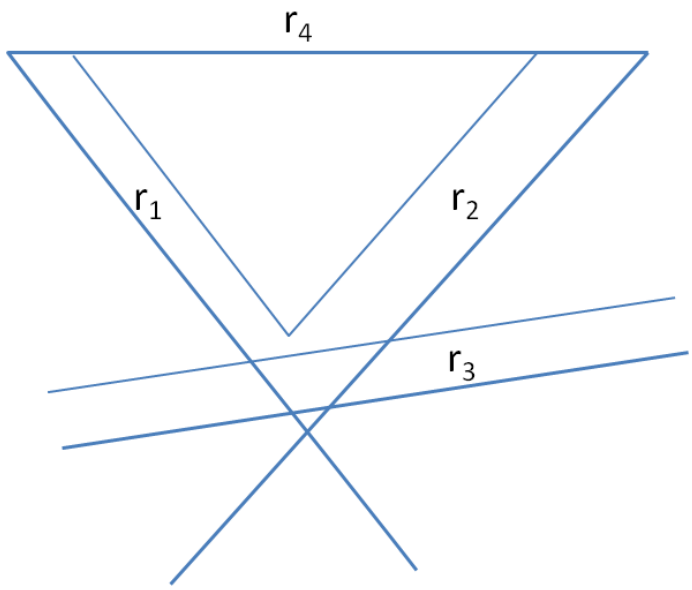


Fig. 4: Polytope bounded by thick hyperplanes r_1, r_2, r_3 and r_4. Note that the topology of the polytope built by the outer hyperplanes (thick) is not the same as the one by the inner hyperplanes (thin).

In the following, we demonstrate the different strategies for start basis generation on the example system *mixed.sys*:

$$0.5x_1 = x_2x_3$$
$$0.5x_2 = x_1x_3$$
$$0.5x_3 = x_1x_2$$

on $D_1 = D_2 = D_3 = [-1, 1]$. The basis variables are marked with a box around them. When pivoting with x_{23}, x_{13}, x_{12}, the system is in reduced row-echelon form

$$0.5x_1 = \boxed{x_{23}}$$
$$0.5x_2 = \boxed{x_{13}}$$
$$0.5x_3 = \boxed{x_{12}}$$

and can be completed with two inequalities of the Bernstein polytope for x_{ii}, $i = 1, 2, 3$ into a start basis. Figure 6, left, shows a statistics of the interval width of the objective value in the course of pivoting. The computation as described in [2] performs one reduction of each variable before bisecting the largest interval. For this problem (tableau size $m = 19 \times n = 33$) on $D_1 = D_2 = D_3 = [-1, 1]$, it needs 47 reductions (410 pivot steps in total), 10 bisections, and the solution time is 0.30s using the sparse form (Windows 7/Visual Studio 2008, Intel Pentium P8700, Dual Core 2.53GHz). The same solving using the dense form takes 0.82s and produces different error widths. The largest condition number of the tableau is 2.0.

When pivoting with x_1, x_{13}, x_{12}, the system is in reduced row-echelon form

$$0.5\boxed{x_1} = x_{23}$$
$$0.5x_2 = \boxed{x_{13}}$$
$$0.5x_3 = \boxed{x_{12}}$$

and can be completed with three inequalities of the Bernstein polytope x_{23} into a start basis. Note that the reduced row-echelon form for variables x_2 and x_3 is similar and thus omitted here. Figure 6, right, shows a statistics of the interval width of the objective value in the course of the pivot steps. For this problem, the computation performs 47 reductions (275 pivot steps in total), 10 bisections, and the solution time is 0.32s. The same solving using the dense form takes 0.72s. The worst condition number of the tableau is 846.3, and it results in larger objective value intervals, i.e., worse lower bounds. In the comparison, the second start basis results in less pivot steps in the dual simplex iteration. Tableaus of larger condition number normally occur if very small pivots were chosen. This can be necessary for Bernstein inequalities corresponding to very small intervals D_i. It is possible to replace such a Bernstein polytope by a thick plane or a thick line as described in [2].

In general, the tableau method tends to populate rows quickly. On the system *mixed.sys*, the user and Bernstein region of the tableau get filled approximately 60% to 80%.

For comparison with the revised simplex implementation *SoPlex*, we compute a rigorous lower bound $b^t y + e$ using the duality gap

$$e = \min\{(c^t - y^{*\,t}A)x : x \in D\}$$

The primal solution vector x^* is directly available, and the corresponding dual solution vector y^* can be derived from the constraint slackness at x^*. When solving the system *mixed.sys*, the code performs around 500 pivot steps, which are fast due to the revised simplex implementation. But large duality gap sizes (larger than 10^{-10}) occur for linear programs, where no bound reduction could be achieved due to an early termination in *SoPlex*.

 ISBN: 978-1-61208-290-5

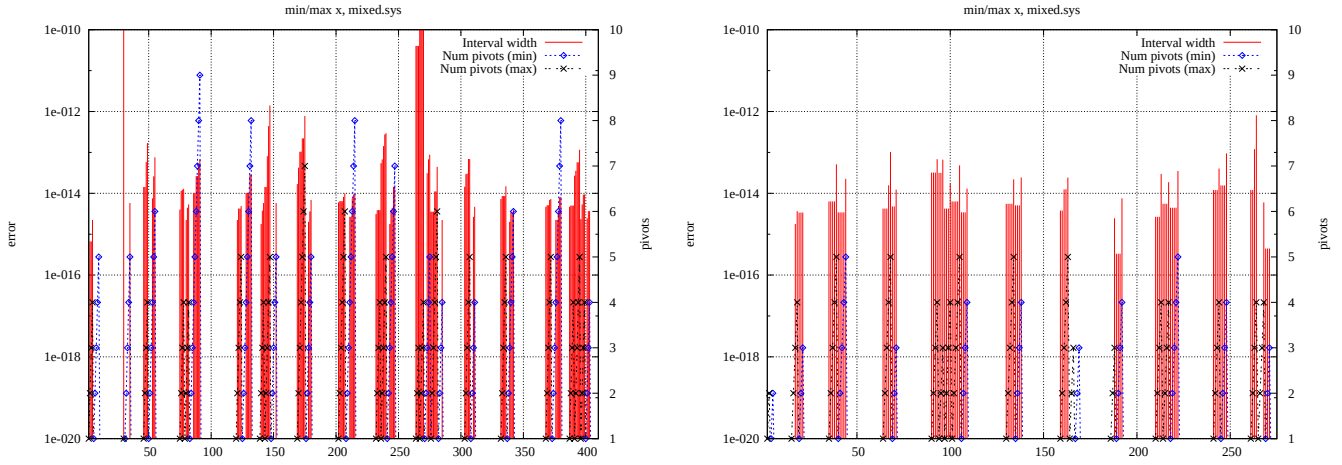


Fig. 6: System *mixed.sys* with start basis from Bernstein polytope for x_{ii} (left) and from the equation system itself (right). Errors are derived from the interval width of the objective value.

IV. Conclusion

In this paper, we presented a new way to implement the dual simplex algorithm in tableau form with pivot steps using interval arithmetic for the direct computation of a reliable lower bound. Such an algorithm can be used for example in a polynomial system solver using linear relaxations. Compared to a lower bound computed from the duality gap, it is not so much affected by the condition number of the given linear program and an early termination of the simplex code. But due to the use of the tableau form it performs more floating-point operations than a revised simplex implementation (e.g., SoPlex [5]) for a sparse input system due to the gradual increase of non-zeros during pivot steps.

The computation is based on the Gauss-Jordan algorithm and performs pivot steps in interval arithmetic that are parallelizable with OpenMP. We always avoid critical sections for reduction operations. In the steepest edge pricing rule and the ratio test, a suboptimal choice does not produce wrong results and we could not observe any performance degradation. For tableau storage, we choose row major order, which avoids recomputing the row factors for a row operation. We prefer sparse storage (as rows of index/entry pairs) over dense (as an array). To avoid further fill-in, we make use of a separate basis description so that the locations of unit vectors in the tableau are known.

For large systems, it is possible to work with an active set of variables $\{x_j\}$ and replace all others with their interval D_j, which is a major benefit of using an interval right-hand side. But with such wide intervals the topology changes, outlined in Figure 4, need to be handled.

Acknowledgements

This research work has been funded by NPRP grant number NPRP 09-906-1-137 from the Qatar National Research Fund (a member of The Qatar Foundation).

References

[1] R. Kearfott, “Discussion and empirical comparisons of linear relaxations and alternate techniques in validated deterministic global optimization,” *Optimization Methods and Software*, vol. 21, no. 5, 2006, pp. 715–731.

[2] Ch. Fünfzig, D. Michelucci, S. Foufou, “Nonlinear systems solver in floating-point arithmetic using LP reduction,” in *ACM/SIAM Symposium on Solid and Physical Modeling*, 2009, pp. 123–134.

[3] M. Bodrato, A. Zanoni, “Intervals, syzygies, numerical Gröbner bases : A mixed study,” in *CASC 2006 Proceedings*, V. G. Ganza, E. W. Mayer, and E. V. Vorozhtsov, Eds., vol. 4194. LNCS, Springer, September 2006, pp. 64–76.

[4] J. Verschelde, “Polynomial homotopy continuation with PHCpack,” *ACM Communications in Computer Algebra*, vol. 44, no. 4, 2010, pp. 217–220.

[5] R. Wunderling, “SoPlex library version 1.4.2,” 1996.

[6] C. Keil, “A comparison of software packages for verified linear programming,” *Preprint Institute of Reliable Computing, Hamburg University of Technology*, 2008.

[7] ——, “Lurupa – rigorous error bounds in linear programming,” in *Algebraic and Numerical Algorithms and Computer-assisted Proofs, Dagstuhl Seminar Proceedings (Number 05391)*, B. Buchberger, S. Oishi, M. Plum, and S. Rump, Eds., July 2006.

[8] J. Hall, “Towards a practical parallelisation of the simplex method,” *Computational Management Science*, vol. 7, no. 2, 2010, pp. 139–170.

[9] R. E. Bixby, A. Martin, “Parallelizing the dual simplex method,” *INFORMS Journal on Computing*, vol. 12, no. 1, Jan. 2000, pp. 45–56.

[10] C. Papadimitriou, K. Steiglitz, *Combinatorial optimization: Algorithms and Complexity*. Dover, 1998.

[11] R. Bixby, “Solving linear and integer programs,” in *Block Course Combinatorial Optimization at Work, Berlin*, M. Grötschel, Ed., 2009.

[12] S. F. McGinn, R. E. Shaw, “Parallel Gaussian Elimination using OpenMP and MPI,” in *16th Annual Int. Symp. on High Performance Computing Systems and Applications, Moncton, Canada*, June 2002, pp. 169–176.

 ISBN: 978-1-61208-290-5

Implementing a Generalized Cobb Model for Production Functions

Alina Andreica
IT Department
Babes-Bolyai University, Cluj-Napoca, Romania
alina.andreica@ubbcluj.ro

Florina Covaci
IT Department
Babes-Bolyai University, Cluj-Napoca, Romania
florina.covaci@ubbcluj.ro

***Abstract* — The paper proposes an extension of Cobb-Douglas model for production functions applicable in multi-product and regional contexts and a combined symbolic & numeric method for solving the system. In this respect, we propose a new application of Buchberger's algorithm for computing Gröbner basis in simplifying the polynomial set that is obtained by modelling economic production systems. We present our Mathematica categorical implementation of Gröbner basis algorithm that can be applied for performing necessary computations. We apply Gröbner basis algorithm for some specific production function cases proposed in the economic literature and discuss the results. We consider that Gröbner basis algorithm is important in processing (according to various variable orderings) the polynomial set that is constructed and in synthesizing the most important economic characteristics taken into account by the model. Further on, numeric methods can be applied if necessary. We also note the importance of generic implementations of Buchberger's algorithm, which can be easy adapted for various domains.**

Keywords-Cobb-Douglas model; generalized production functions; symbolic modelling; Buchberger algorithm; generic implementations.

I. INTRODUCTION AND WORKING FRAMEWORK

Category theory in symbolic computation introduces techniques for implementing general working contexts of performing symbolic algorithms. The defined categories can later be particularized for various domains, by elegantly using the same definition. Definitions for categories and domains, from the symbolic point of view, can be found in [1], [2]. In these papers, we describe an extension of Mathematica – a computer algebra or symbolic computation system [15] – with a new type system, containing algebraic categories and abstract definitions of Gröbner basis [4] simplification algorithm. Within this package, the algorithms that implement Buchberger's method for computing the Gröbner basis and the reduced Gröbner basis for a given set of polynomials are applied for multivariate polynomial domains over various coefficient rings.

Buchberger's algorithm for Gröbner basis [4] has found numerous applications in various fields related to polynomial simplification over various fields [10], [11] including computational geometry [12].

Within this paper, we present a new application of Buchberger's algorithm for Gröbner basis, belonging to the economic field, namely, for simplifying production function sets according to Cobb-Douglas model [7].

Section 2 describes the Cobb-Douglas model for economic production functions [7]. Section 3 presents the basic principles of Buchberger algorithm for computing the Gröbner basis. In Section 5, we present the model we propose for generalizing the Cobb Douglas model at a macroeconomic scale, taking into account many countries. In Section 6, we describe the basic principles of categorical polynomial definition and processing from our Mathematica implementation, while Section 7 presents our implementation of computing the Gröbner basis. Conclusions reveal the most important contributions of the paper.

II. COBB-DOUGLAS MODEL FOR ECONOMIC PRODUCTION FUNCTIONS

In economic modelling, the Cobb-Douglas functional form of production functions [7] is widely used to represent the relationship of a production output in respect with specific inputs. The model was proposed by Knut Wicksell [14], and tested against statistical evidence by Paul Douglas and Charles Cobb in 1928 [7].

The model states [7] that a production function can be written in the functional form:

$$Y = AL^{\alpha}K^{\beta} \tag{1}$$

where:

Y the production output

L represents the labour input

K represents the capital input

A, α and β are constants determined by technology. The exponents α and β are output elasticity coefficients with respect to labour and capital, respectively. Output elasticity measures the responsiveness of output to a change in levels of either labour or capital used in production [7].

According to the original model [7], the following cases are considered relevant:

- if $\alpha+\beta=1$, the production function has constant returns to scale;
- if $\alpha+\beta<1$, returns to scale are decreasing;
- if $\alpha+\beta>1$ returns to scale are increasing. Assuming perfect competition, α and β can be shown to be labour and capital's share of output.

Cobb and Douglas were influenced by statistical evidence that appeared to show that labour and capital shares of total output were constant over time in developed

 ISBN: 978-1-61208-290-5

countries; they explained this feature by applying statistical fitting in least-squares regression of their production function [7].

III. BUCHBERGER'S ALGORITHM

Buchberger's algorithm computes the Gröbner basis [4] of a given set of polynomials over a coefficient ring as a "simplified" polynomial set; the Gröbner basis G for a polynomial set F is a "reduced" form of G [4], [5] that generates the same ideal as F.

The "simplified" polynomial set that is obtained in the reduction process is significant in solving various problems involving polynomial sets. Buchberger synthesizes [5] a variety of application fields for Gröbner basis algorithm, revealing its importance in systems theory. The application presented in this paper is founded on the principles presented in [5].

Based on category theory and generic principles, recent versions of Buchberger's algorithm [6] are abstract implementations, which offer generic frameworks for applying the algorithm. The implementation we presented for Mathematica in [1] and [3] are based on these generic principles, proving to have important flexibility and extendibility advantages in applying the algorithm over various domains

Present built-in Mathematica implementations of Buchberger algorithm support polynomials over the following coefficient domains: InexactNumbers, Rationals, RationalFunctions and Polynomials[*x*] [16].

IV. GENERALIZED COBB-DOUGLAS MODEL. APPLYING BUCHBERGER'S ALGORITHM

We generalize the Cobb-Douglas model for production functions in order to represent production states at a macroeconomic scale. We consider that such a model is relevant in order to better grasp the production phenomena that appear in economies.

In this respect, we propose a set of Cobb-Douglas functions for expressing the necessary labour input and capital input to obtain a certain output within an economy. Taking into consideration the economy of a country, we consider the variables Lp, Kp as the labour input, respectively capital input for obtaining the product p and Gross Domestic Product (GDP) as the global output. For more economic regions, Gross Domestic Products (GDPs) can be represented as:

$Y_i, \mathrm{i} = 1, \mathrm{m}$, - Y_i being the Gross Domestic Product (GDP) generated by region i. If we consider that each region i produces n_i products, we arrive to the generalized model (2).

We consider that the model we propose is relevant for expressing production characteristics at a macroeconomic scale. Furthermore, the possibility of expressing production characteristics in regional development frameworks, can be very useful in the context of economic European integration.

$$\begin{cases} Y_1 = A_{11} L_{p_1}^{\alpha_{11}} K_{p_1}^{\beta_{11}} + A_{12} L_{p_2}^{\alpha_{12}} K_{p_2}^{\beta_{12}} + \ldots + A_{1n_1} L_{p_{n_1}}^{\alpha_{1n_1}} K_{p_{n_1}}^{\beta_{1n_1}} \\ Y_2 = A_{21} L_{p_1}^{\alpha_{21}} K_{p_1}^{\beta_{21}} + A_{22} L_{p_2}^{\alpha_{22}} K_{p_2}^{\beta_{22}} + \ldots + A_{2n_2} L_{p_{n_2}}^{\alpha_{2n_2}} K_{p_{n_2}}^{\beta_{1n_2}} \\ \quad . \\ \quad . \\ \quad . \\ Y_m = A_{m1} L_{p_1}^{\alpha_{m1}} K_{p_1}^{\beta_{m1}} + A_{m2} L_{p_2}^{\alpha_{m2}} K_{p_2}^{\beta_{m2}} + \ldots + A_{mn_m} L_{p_{n_m}}^{\alpha_{mn_p}} K_{p_{n_m}}^{\beta_{mn_m}} \end{cases} \quad (2)$$

Consequent to defining the functional production polynomials, an important task is to simplify the polynomial set in order to obtain in a canonical form the functions that express labour and capital inputs for different products and regions at a macroeconomic scale. Such a form is important for characterizing the production features in a synthesized manner. In order to obtain this simplified form, we can apply Buchberger's algorithm.

For monomial exponents representing elasticity coefficients that have real values, we can apply the reduction steps by computing the necessary subtractions of the exponent lists: in respect with the order relation on the real set $\Re$, extended to tuples of real exponents, $\Re_n$, the "leading" monomial of each polynomial can be reduced by computing monomials with corresponding exponent subtractions in respect with the other polynomials. For the reduction step in Buchberger's algorithm, instead of taking into account the least common multiple for pairs of exponent vector lists, we compute the maximum of two exponent vector lists in respect with $\Re_n$

In our Mathematica implementation based on generic principles [1], [3], we easily defined the Lcm (Least Common Multiple) operator in the exponent vector package *VectExp* as a Max operator and we supplementary defined the real domain in our coefficient domain package *DomCoef* – for which a code overview is given in Fig. 1:

```
DomReal[R_]:=Module[{},
  InelCom[R,"+","*","0","1"];
  (* Mathematica definition [2] *)
  R["+",a_Real,d_Real]:=a+d;
  R["+",a_Real,Infinity]:=Infinity;
  R["+", Infinity, d_Real]:=Infinity;
  R["-",a_Real,b_Real]:=a-b;
  R["*",a_Real,b_Real]:=a*b;
  R["/",a_Real,b_Real]:=a/b;
  R["=",a__,b__]:=SameQ[a,b];
  R["<>",a__,b__]:=UnsameQ[a,b];
  R["<",a__,b__]:=a<b;
  R["<=",a__,b__]:=a<=b;
  R[">",a__,b__]:=a>b;
  R[">=",a__,b__]:=a>=b;
  R["0"]:=0;
  R["max"]:=Infinity;
  R["/",a_List,b_Real]:=a/b;
  R["+",a__,b__]:=a+b;
  R["-",a__,b__]:=a-b;
  R["*",a__,b__]:=a*b;
  R["/",a_List,b_Real]:=a/b;
  R["/",a__,b__]:=a/b;  ]
```

Figure 1. Mathematica abstract definition for the real domain.

The higher impact of labour L or capital K variables in the above described polynomials – (2) – can be evaluated by taking into account appropriate orderings of L_i, i=1,m, K_i, i=1,m. Moreover, the model can be enhanced by supplementary taking into account new production variables. In this respect, [17] proposes to take into account a third variable as a sum of variables with a smaller production impact. Still, the model we propose may take into account any number of variables.

We applied Buchberger's algorithm for sets of 2-3 polynomials representing GDPs in 2 countries by taking input values and elasticity coefficients proposed in [17].

For cases in which the reduced polynomial set should be further processed, numeric solving methods may be consequently applied

V. POLYNOMIAL CATEGORICAL IMPLEMENTATION

In order to implementing the multivariate polynomial category, we define an auxiliary domain for monomials [1], naming it the exponent vector domain [9].

An exponent vector with a given base of identifiers (for example, x,y,z) will be retained by the list of exponents corresponding to each variable. In operating upon exponent vectors, we use two types of representations: lists, respectively products of primes with the exponents in question [9]. We use the latter representation [1], the exponent vector e_1, e_2, ..., e_n will be retained and processed by the number

$$p_1^{e_1} * p_2^{e_2} * ... * p_n^{e_n}, \quad (3)$$

where p_1, p_2, ..., p_n are prime numbers, for simplifying computations - the first prime numbers. In this representation, an exponent vector sum reduces to the corresponding prime numbers product, whereas the greatest common divisor (Gcd) and the least common multiple (Lcm) can be computed by similar operations upon prime products.

The exponent vector domain is an abelian monoid [2] that introduces the following operations:

- computing neutral, minimum and maximum elements ("0", "Max","Min");
- conversions between the list and number internal forms ("ListaInVectorNr", "VectorNrInLista");
- sum ("+"), greatest common divisor ("Gcd") and least common multiple ("Lcm") for two exponent vectors;
- positiveness test for an exponent vector ("Pozitiv");
- divisibility test for two exponent vectors ("|");
- relational operators ("<", ">", "=", "<=", ">=", "<>") between two exponent vectors - we shall consider the lexicographical ordering corresponding relations are implemented by string[...] functions);
- conversions between external and internal forms ("Inp", "Out").

We give in Fig. 2 an overview of the Mathematica package which defines the exponent vector domain (for the functions in italics we omitted the body). All operations within an exponent vector domain V, created by the function VectExp[V,lv], where lv is the variable list representing the base, will be prefixed with the domain name and will have as the first parameter the operation code. For example, V["+",v1,v2] returns the sum of two exponent vectors (in internal form), V["Gcd",v1,v2] computes the greatest common divisor, while V["Lcm",v1,v2] computes the least common multiple.

```
BeginPackage["VectoriExp‘"]
VectExp::usage="VectExp[V ,lv List] defines the
exponent vector domain V, with the base lv "
Intreg::usage="Integer domain"
string::usage="string operations"
Begin["VectoriExp‘Private‘"]
Needs["HierMath‘"]
string["Rel",s1 String,s2 String]:= Mathematica definition[2]
(*tests the relation <, =, > between s1 and s2, returning -1,0,1*)
(* "Rel" may be "<", ">", "=", "<>" *)
VectExp[V ,lv List]:= Mathematica definition [2] (*creates the
exponent vector domain V, with the base lv*)
MonoidCom[V,"+",en]; Mathematica definition [2]
(*creates an abelian monoid [2]*)
V["ListaInVectorNr",l List]:= Mathematica definition [2]
V["VectorNrInLista",nr Integer]:= Mathematica definition[2]
V["+",l1 List,l2 List]:=l1+l2;
V["-",l1 List,l2 List]:=l1-l2;
V["Pozitiv",l List]:= Mathematica definition [2]
(*tests if all list elements are >0*)
V["Gcd", l1 List, l2 List]:= Mathematica definition [2]
(*computes greatest common divisor *)
V["Lcm", l1 List, l2 List]:= Mathematica definition [2]
(computes teast common multiple *)
V["Rel", l1 List, l2 List]:= Mathematica definition [2] (*tests the
relation <, =, > between l1 and l2, returning -1,0,1*)
V["|",l1 List,l2 List]:= Mathematica definition[2]
(* divisibility test *)
V["Out",l List]:= Mathematica definition [2]
(* output form *)
V["Inp",e ]:= Mathematica definition [2] (*transforms an input with
the syntax x[^e1]*y[^e2]... into the internal list form;
the code is rather complex and based on Mathematica
internal forms*)
V["Max"]:=max; V["Min"]:=min;]
End[]
EndPackage[]
```

Figure 2. Mathematica abstract definition for the exponent vector domain.

The representation of a polynomial (polinom.m package) uses a list of two elements: the exponent vector list, lexicographically ordered, and the corresponding coefficient list [2]. For example, the polynomial 2*xˆ2*z-5*y (with the base {x,y,z}) will be represented as {{{0,1,0},{2,0,1}}, {-5,2}}

We give below the main part of the Mathematica package which defines the polynomial category (polinom.m package [2]; we omitted the bodies for the functions in italics). Within Mathematica definitions, functional and parametric specification of operations within various domains can be noticed (Figs. 1, 2).

 ISBN: 978-1-61208-290-5

For example, within the polynomial domain Pol, defined by Polinom[Pol, DCoef, DVect, l], where DCoef is the coefficient domain and DVect is the exponent vector domain with the base l, the construction DVect["|",v,Pol["Monom",i,P]] performs a divisibility test - within DVect exponent vector domain - between two exponent vectors, the second one corresponding to a monomial selected from a polynomial P (by "Monom" operation, within the polynomial domain Pol). DVect["+",o1,o2] is the sum of o1 and o2 in DVect domain, whereas DCoef["+",o1,o2] is a sum in DCoef domain. An overview of the Gröbner basis package we have defined [1], [2] is given in Fig. 3:

```
BeginPackage["Polinom`"]
Polinom::usage="Polinom[Pol,DCoef,DVect,l] defines the
multivariate polynomial domain Pol, over the coefficient domain
DCoef and the exponent vector domain DVect (of monomials), with
the basis l"
Begin["Polinom`Private`"]
Needs["VectoriExp`","DomCoef`"]
Polinom[Pol ,DCoef ,DVect ,baza List]:=Module[n,lv,
(*defines a multivariate polynomial domain, with coefficients in
DCoef, over the exponent vector domain Dvect, with the base
baza*)
InelCom[DCoef,"+","*"]; (*creates an abelian ring [2]*)
VectExp[DVect,baza];
Pol["DomCoef"]:=DCoef;
Pol["DomVectExp"]:=VectExp;
Pol["Init"]:=List[List[DVect["Max"]],List[]];
(*returns the null polynomial, with a sentinel in the exponent vector
list*)
Pol["Nul",P ]:=Mathematica definition [2] ;
(* returns True if P is null *)
Pol["Nr",P ]:=Length[P[[1]]]-1; (*dimension*)
Pol["Monom", i ,P ]:=P[[1]][[i]];
(*the ith monomial from the polynomial P*)
Pol["Coef",i ,P ]:=If[i<=Pol["Nr",P],P[[2]][[i]],0];
Pol["Indice",l ,P ]:= Mathematica definition [2]
(*the index of the monomial l in the polynomial P*)
Pol["MonomGrMax",P ]:=P[[1]][[Pol["Nr",P]]];
Pol["CoefGrMax",P ]:=P[[2]][[Pol["Nr",P]]];
Pol["AdaugMonom",T ,v ,c ]:= Mathematica definition [2]
 (*adds to the polynomial T the monomial formed by the exponent
vector v and the coefficient c taking into account the lexicographical
ordering; if v exponent vector exists, it adds the coefficient c to the
appropriate existing one*)
Pol["MonomInPol",l :List,c :Number]:= Mathematica definition [2]
(*transforms a monomial into the equivalent polyniomial *)
Pol["+",P1 ,P2 ]:= Mathematica definition [2]
(* returns the sum of P1, P2 *)
Pol["*",P1 :List,P2 :List]:= Mathematica definition [2]
(* returns the product of P1, P2 *)
Pol["&",nr :Number,P :List]:= Mathematica definition [2]
(*multiplies P by the number n*)
Pol["-",P1 ,P2 ]:=Module[{P}, (*polynomial subtraction *)
P=Pol["&",-1,P2]; Return[Pol["+",P1,P]]; ];
Pol["/",P ,v List,c :Number]:= Mathematica definition [2]
(* divides each of P's monomials by the exponent vector v and
coefficient c and returns the result *)
Pol["|",v List,P ]:= Mathematica definition [2]
(* tests whether the exponent vector v divides any of P's monomials
and returns True or False *)
Pol["Out",P ]:= Mathematica definition [2] (*polynomial display*)
Pol["Inp",e ]:= Mathematica definition [2] (*transforms an input
polynomial into the internal form; the code is rather complex and
based on Mathematica internal forms*)
] (*Module*)
End[]
EndPackage[]
```

Figure 3. Mathematica abstract definition for the polynomial domain.

The following section is dedicated to the Gröbner basis algorithm and its application to production functions.

VI. GROEBNER BASIS ABSTRACT IMPLEMENTATION AND APPLICATION CASES FOR PRODUCTION FUNCTIONS

We implemented Buchberger's algorithms for computing the Gröbner basis and the reduced Gröbner basis [4] of a polynomial set into Mathematica packages: groebner.m and groebred.m [1], [2]. The functions which compute the Gröbner bases are parameterized with a polynomial domain, therefore they can be applied for polynomial domains over any consistent coefficient domain that is previously defined. Note that a polynomial domain is created by using the polynomial categorical definition within polinom.m package, which is parameterized with a coefficient domain defined in domcoef.m package [1], [2]. Within groebner.m package [1] we implemented Buchberger's Gröbner basis algorithm [4] - BazaGroebner[] function. We completely described the algorithmic iterations for computing the normal form of a polynomial modulo a polynomial set - Normal[Pol,F,g] function, where Pol is the current polynomial domain. For computing the S-polynomial of two polynomials, we implemented the formula proposed in [9] - SPol[] function.

An overview of the Gröbner basis package we have defined [1], [2] is given in Fig. 4:

```
BeginPackage["Groebner`"]
Normal::usage="Normal[Pol,F,g] verifies if g is in normal form mod
F, over the polnomial domain Pol"
FormaNormala::usage="FormaNormala[Pol,DCOef,DVect,F,p]
returns the p's normal form modulo F; operations are
performed over the polynomial domain Pol"
SPol::usage="SPol[Pol,DCoef,DVect,P1,P2] computes
Rez=SPol(P1,P2), in the polynomial domain
Pol(DCoef,DVect)"
BazaGroebner::usage="BazaGroebner[Pol,DCoef,DVect,F] returns,
for F set of polynomials over the domain
Pol(DCoef,DVect), F's Groebner base"
MultPolExtInInt::usage="MultPolExtInInt[Pol,M] transforms a set of
polynomials in external representation into internal representation
(operations over Pol domain)"
MultPolIntInExt::usage="MultPolIntInExt[Pol,M] transforms a set of
polynomials in internal representation into external representation
(operations over Pol domain)"
Tiparire::usage="prints a set of polynomials given in internal
representation"
TipPerechi::usage="prints a set of polynomial pairs given in internal
representation"
Begin["Groebner`Private`"]
Needs["Polinom`"]
PolNormal[Pol ,F List,g ]:= Mathematica definition [2]
(*Verifies whether g is in normal form mod F, i. e. no
```

```
monomial of g is divisible by the leading monomial of
any polynomial belonging to F - set of polynomials.
Operations are performed within the polynomial domain Pol*)
FormaNormala[Pol ,DCoef ,DVect ,F List,p ]:= ]:= Mathematica
definition [2]
(*Returns p's normal form mod F; operations are performed
within the polynomial domain Pol(DCoef,DVect) *)
SPol[Pol ,DCoef ,DVect ,P1 ,P2 ]:= Mathematica definition [2]
(*computes, within the polynomial domain P, Rez=SPol(P1,P2)*)
MultPolIntInExt[Pol ,M List]:= ]:= Mathematica definition [2]
(*transforms M polynomial set from internal form into a set of
external forms*)
MultPolExtInInt[Pol ,M List]:= Mathematica definition [2]
(*transforms M polynomial set from external form into a set of
internal forms *)
Tiparire[Pol ,M List]:= Mathematica definition [2] (*displays a
polynomial set*)
TipPerechi[Pol ,M List]:= Mathematica definition [2]
(*displays a set of polynomial pairs*)
BazaGroebner[Pol ,DCoef ,DVect ,baza List,M List]:= Mathematica
definition [2]
(* returns the Groebner basis of the polynomial set M
(given as a list), within the polynomial domain Pol *)
End[]
EndPackage[]
```

Figure 4. Mathematica abstract definition for the Gröbner basis algorithm.

Groebred.m package [1], [2] implements Buchberger's algorithm for computing the reduced Gröbner basis [4].

Comparing our implementation to the Mathematica built in one GroebnerBasis[{$poly_1$, $poly_2$, …},{x_1, x_2, …}], our abstract based one is obviously slower, but it enables computations over various abstract coefficient domains, which can be defined. The built in GroebnerBasis implementation works over rational numbers, integers, rational functions or inexact numbers, with additional computation options [16].

We have applied the Gröbner basis algorithm in Mathematica for various cases of production functions proposed for Romania and Moldova for the period 2002-2004 in [17]. We have denoted with R the polynomial corresponding to Romania's Gross Domestic Product (GDP) function for this period and with M – the polynomial corresponding to Moldova's Gross Domestic Product (GDP) function for the same period.

For the set of polynomial production functions:

$R = 37.4^a\ 62.6^b\ X+45.3^a\ 54.7^b\ Y+49.1^a\ 50.9^b\ Z$

$M = 33.8^a\ 66.2^b\ X+33.4^a\ 66.6^b\ Y+40.5^a\ 59.5^b\ Z$

where X, Y, Z are technology based variables, and using symbolic elasticity coefficients a, b [17], the Gröbner basis still contains two polynomials in X, Y, Z:

$\{-1531.14^a\ 3621.14^b\ Y+1249.16^a\ 4169.16^b\ Y-1659.58^a\ 3369.58^b\ Z+1514.7^a\ 3724.7^b\ Z,33.8^a\ 66.2^b\ X+33.4^a\ 66.6^b\ Y+40.5^a\ 59.5^b\ Z,37.4^a\ 62.6^b\ X+45.3^a\ 54.7^b\ Y+49.1^a\ 50.9^b\ Z\}$

Using the α, β elasticity proposed in [17] as a, b values, we obtain the liniar polynomials:

$R = 45.3457\ X+48.6099\ Y+49.7656\ Z$

$M = 43.4613\ X+43.2359\ Y+46.7666\ Z$

The GroebnerBasis function reduces the X variable (corresponding to year 2002) from the R polynomial and the Y variable (corresponding to year 2003) from the M polynomial, generating the following simplified set:

$\{1.\ Y+0.27755\ Z,1.\ X+0.799942\ Z\}$

We may infer that during the period 2002-2004, for Romania and Moldova, the most relevant evolution years from the production point of view were 2003 and 2004 for Romania and 2002 and 2004 for Moldova. Such results have to be correlated with other macroeconomic variables.

Another possible application case would be the one taking into account countries from Latin America, considering the K values as capital flows and the L values given in [18]. Simplified polynomials would mean similar evolutions in different countries.

VII. CONCLUSION AND FUTURE WORK

We addressed a problem from an economic field, namely a generalized model for production functions, by applying computer algebra tools.

We generalized Cobb-Douglas model for production functions in multi-product and regional contexts by constructing a representative polynomial set in respect with the production inputs (labour, capital, other variables) and we propose the application of Buchberger's algorithm in simplifying the polynomial set that is obtained. We consider that Gröbner basis algorithm is important in processing (according to various variable orderings) the polynomial set that is constructed and in synthesizing the most important economic characteristics taken into account by the model.

We presented our Mathematica categorical implementation of Buchberger's algorithm for Gröbner basis algorithm that can be applied for performing necessary computations. We underline the importance of such abstract implementations, which can be easy adapted for various domains based on parameterized principles. Our implementation is actually an extension of Mathematica with a type system.

We applied the Gröbner basis algorithm for Gross Domestic Product (GDP) functions of Romania and Moldova for the period 2002-2004 using data proposed by Zaman et al in [17] and we discuss the results of applying Buchberger's simplification algorithm on these polynomial sets. Similar processings can be performed for other countries, using specific values that are available in economic analyses for the input data in the production functions.

We intend to further work on the proposed model and to study other cases from the economic literature. We also intend to extend our implementation of Buchberger algorithm for new domains, based on the same abstract principles.

REFERENCES

[1] A. B. Andreica, "Parameterized Types for Categorical Definitions in Mathematica", Symbolic and Numeric Algorithms for Scientific Computing - SYNASC 2002 International Workshop, Mirton, Editors: D. Petcu, V. Negru, D. Zaharie, T. Jebelean, 2002, pp. 8-25.

[2] A. B. Andreica, "Defining Algebraic Categories in Mathematica", Analele Universitatii de Vest Timisoara, Seria Matematica - Informatica, Categ CNCSIS B+, XLI, 2003, pp. 9 - 23

 ISBN: 978-1-61208-290-5

[3] A B. Andreica, "Implementing Parameterized Type Algorithm Definitions in Mathematica", Eighth International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, Lisa O'Connor editor, IEEE Computer Society Press, 2006, pp. 1-6.

[4] B. Buchberger, "Gröbner Bases: An Algorithmic Method in Polynomial Ideal Theory", in Mathematics and Its Applications, Multidimensional Systems Theory, N. K. Bose Ed., D. Reidel Publishing Co., 1985, chapter 6.

[5] B. Buchberger, "Gröbner Bases and Systems Theory", Multidimensional Systems and Signal Processing, 12, Kluwer Academic Publishers, 2001, pp. 223-251.

[6] B. Buchberger, "Towards the Automated Synthesis of Groebner Bases Algorithm", RACSAM, vol Falta, 2004, pp.1-10.

[7] C W. Cobb and P H Douglas, "A Theory of Production", American Economic Review, 18, 1928, pp. 139-165.

[8] D. Gruntz and M. Monagan, "Introduction to Gauss", MapleTech, Birkhuser, 9, 1993, pp. 23-35.

[9] D. Gruntz, "Gröbner Bases in Gauss", MapleTech, Birkhuser, 9, 1993, pp. 36-46.

[10] K. Iwancio and M. Singer, "Applications of Groebner Bases", http://www4.ncsu.edu/~kmiwanci/app_gbases_2.pdf [retrieved: 05, 2013]

[11] V. Levandovskyy, "Non–commutative Computer Algebra for polynomial algebras: Groebner bases, applications and Implementation", http://d-nb.info/976594358/34 [retrieved: 05, 2013]

[12] V. Powers and B. Reznick, "A new bound for Pólya's Theorem with applications to polynomials positive on polyhedra", Journal of Pure and Applied Algebra, Vol. 164, Issues 1–2, Oct 2001, pp. 221–229 http://www.sciencedirect.com/science/article/pii/S0022404900001559 [retrieved: 05, 2013]

[13] C. Ratiu-Suciu, F. Luban, D. Hincu, and N. Ene, Modelarea si simularea proceselor economice , Biblioteca electronică a Academiei de Studii Economice Bucureşti, [retrieved: 05, 2013]
http://www.ase.ro/biblioteca/carte2.asp?id=70&idb=7

[14] B. Sandelin, "The Early Use of Wicksell-Cobb-Douglas Function: A Comment on Weber", Journal of the History of the Economic Thought, Vol. 21, Issue 02, Cambridge University Press, June 1999, pp. 191-193, doi: http://dx.doi.org/10.1017/S105383720000314X.

[15] S. Wolfram, Mathematica, 1992.

[16] Mathematica Online Tutorial [retrieved: 06, 2013] http://reference.wolfram.com/mathematica/guide/Mathematica.html

[17] G. Zaman, Z. Goschin, I. Partachi, and C. Herteliu, "The Contribution of Labour and Capital to Romania's and Moldova's Economic Growth", Journal of Applied Quantitative Methods, Vol. 2, Issue 1, March 30, 2007, http://jaqm.ro/issues/volume-2,issue-1/pdfs/zaman_goschin_partachi_herteliu.pdf [retrieved: 06, 2013]

[18] Capital flows and labour costs values for countries in 2013 http://www.tradingeconomics.com/country-list/capital-flows, [retrieved: 09, 2013]

 ISBN: 978-1-61208-290-5

Trading Redundant Work Against Atomic Operations On Large Shared Memory Parallel Systems

Rudolf Berrendorf
Computer Science Department
Bonn-Rhein-Sieg University
Sankt Augustin, Germany
e-mail: rudolf.berrendorf@h-brs.de

***Abstract*—Updating a shared data structure in a parallel program is usually done with some sort of high-level synchronization operation to ensure correctness and consistency. However, underlying synchronization instructions in a processor architecture are costly and rather limited in their scalability on larger multi-core/multi-processors systems. In this paper, we examine work queue operations where such costly atomic update operations are replaced with non-atomic modifiers (simple read+write). In this approach, we trade the exact amount of work with atomic operations against doing more and redundant work but without atomic operations and without violating the correctness of the algorithm. We show results for the application of this idea to the concrete scenario of parallel Breadth First Search (BFS) algorithms for undirected graphs on two large NUMA shared memory system with up to 64 cores.**

***Keywords*—atomic instructions, redundant work, parallel BFS**

I. INTRODUCTION

Updating a shared data structure in a parallel program as for example an insert operation on a work queue is usually done on an application level with some sort of high-level atomic update operation (e.g., in OpenMP [1] lock-protected, atomic operation, etc.; see [2] [3] for a general discussion). The implementation of such a high-level synchronization operation itself is done by the compiler or inside a runtime system with one or even more atomic instructions (atomic-add, test-and$-\Phi$, compare-and-swap, etc.) of the underlaying processor architecture. The general problem with such atomic instructions is that they are rather costly compared to an ordninary memory access and not really scalable on larger systems [4] [5] (see also section IV for our own investigations on that). The time for *one* such atomic instruction increases significantly under contention as the number of cores in a multi-core/multi-processor system gets larger.

As the use of such synchronized updates on shared data guarantees correct operations on that data, this strict enforcement is often not really necessary. An example is a work queue, where working threads insert new items and idle threads remove items to be worked on. But for certain algorithmic scenarios (e.g., within a certain program phase), a work item may be inserted even multiple times without violating the correctness of the algorithm, but only causing additional redundant work to be done. In such cases, the costly synchronized access can be completely removed for the cost of eventually additional work to be done.

An example for such a scenario is a Breadth First Search (BFS) for undirected graphs (see section III for details). Most of the published parallel BFS algorithms iterate over a vertex frontier where the vertices of the current vertex frontier insert new unvisited vertices to the following vertex frontier. In this scenario, adding a vertex twice in such a frontier generates more work to be done in the next level iteration but does not influence the correctness of the algorithm. Another, more general scenario is the development of asynchronous algorithms [6].

In this paper, we examine such a general strategy for a concrete parallel BFS algorithm on large shared memory multi-core multi-processor systems with up to 64 cores. We examine, what the factors are that influence the amount of additional work, what the amount of additional work is, and whether this additional work without any synchronized access to the work queue trades off against the traditional sychronized access to a work queue doing exactly the amount of work that is necessary.

The paper is organized as follows. After this introduction, we start with an overview of related work, followed by a brief overview on parallel BFS algorithms. After that, we present our new approach, describe our experimental setup, and then evaluate the new approach against the traditional way.

II. RELATED WORK

There are several papers on certain aspects on the optimization of synchronization constructs in a wider sense. This includes, amongst others, reducing the number of consecuting mutex lock/unlocks [7] in a program and compiler optimizations for read/write barriers [8]. Furtheron, there are advanced synchronization techniques trying to minimize synchronization costs including RCU (Read-Copy-Update) [9], special monitors [10], read-writer optimizations [11], and specialized lock-free data structures (e.g., [12]). [2] gives an overview of different aspects on related topics. [13] shows a similar benign race as ours in a parallel BFS algorithm, but without analyzing the influence of that.

An interesting general approach to handle possible concurrent accesses to shared data structures is the concept of transactional memory (original paper [14]). This approach has some similarities with our approach as both are optimistic: do a read-modify-write operation without a critical section and react only is something went wrong. The idea with transactional memory as well as in our approach is that the bad thing happens rather seldom. Transactional memory detects the problem and (depending on the API in use) rolls back the whole transaction and restarts the operation. We instead ignore the problem (and do not even detect the problem) and have more work to do in the future.

 ISBN: 978-1-61208-290-5

III. PARALLEL ALGORITHMS FOR BFS

In our application scenario for the examination, we are interested in undirected graphs $G = (V,E)$ where V is a set of vertices $v_1,...,v_n$ and E is a set of edges $e_1,...,e_m$. An edge e is given by an unordered pair $e = (v_i,v_j)$ with $v_i,v_j \in V$. The number of vertices of a graph will be denoted by $|V| = n$ and the number of edges is $|E| = m$.

Assume a connected graph and a source vertex $v_0 \in V$. For each vertex $u \in V$ define $depth(u)$ as the number of edges on the shortest path from v_0 to u, i.e., the edge distance from v_0. With $depth(G)$ we denote the depth of a graph G defined as the maximum depth of any vertex in the graph relative to the source vertex.

The problem of Breadth First Search (BFS) for a given graph $G = (V,E)$ and a source vertex $v_0 \in V$ is to visit each vertex in a way such that a vertex v_1 must be visited before any vertex v_2 with $depth(v_1) < depth(v_2)$. As a result of a BFS traversal, either the level of each vertex is determined or a (non-unique) BFS spanning tree with a father-linkage of each vertex is created. Both variants can be handled by BFS algorithms with small modifications and without extra computational effort. The problem can be easily extended and handled with directed or unconnected graphs. A sequential solution to the problem can be found in textbooks, based on a queue where all non-visited adjacent vertices of a visited vertex are enqueued. The computational complexity is $O(|V + |E|)$.

If one tries to design a parallel BFS algorithm, different challenges might be encountered. As the computational density of BFS is rather low, BFS is bandwidth limited for large graphs and therefore memory bandwidth has to be handled with care. For a similar reason in ccNUMA systems, data layout and memory access should respect processor locality. In multicore multiprocessor systems, things get even more complicated, as several cores share higher level caches and NUMA-node memory, but have private lower-level caches.

```
 1: function BFS(graph g, vertex source)
 2:   var
 3:     d, distance vector of size |V|. Initial values: ∞
 4:     current, next, vertex container. Initially empty
 5:   end var
 6:   d[source] ← 0
 7:   current.insert(source)
 8:   while current is not empty do
 9:     for all v in current do
10:       for all neighbours w of v do
11:         old = CompareAndSwap(d[w], ∞, d[v] + 1)
12:         if old = ∞ then
13:           next.insert(w)
14:         end if
15:       end for
16:     end for
17:     Barrier
18:     swap current with next
19:   end while
20:   return d
21: end function
```

Fig. 1: Parallel BFS with an atomic CAS-operation

In BFS algorithms housekeeping has to be done on visited / unvisited vertices with several possibilities how to do that. Some of them are based on special container structures for vertex frontiers where information has to be inserted and deleted. Scalability and administrative overhead of these containers are of interest. Generally speaking, these approaches deploy two identical containers (current frontier, next frontier) whose roles are swapped at the end of each level iteration. Fig. 1 shows this in a rather straightforward version with an atomic Compare-And-Swap (CAS) operation in an inner loop (line 11) to detect and update unvisited vertex neighbors. In this atomic operation, a vertex is checked wether it is visited already ($d[w] \neq \infty$), and if not, marks the vertex as visited. Based on this knowledge, only an unvisited vertex gets inserted into the next vertex frontier. After all vertices in the current container are visited, all threads wait at a barrier before work on the next container / frontier gets started (level iteration). This version can be further optimized using chunked lists for every thread. The insert operation of a new vertex into a thread-local chunk can be done in a non-atomic way. But the construction of a global list from thread-local chunks (i.e., the insertion of each chunk into a global list) must still be done in a synchronized way. But as this is done only if a chunk gets full, this is not the critical operation of this algorithm but the detection of visitedness in line 11. Container centric approaches are eligible for dynamic load balancing but are sensible to data locality on ccNUMA systems. Container centric approaches for BFS can be found in some parallel graph libraries [15] [16]. [17] contains an overview and evaluation of several parallel BFS algorithms.

For level synchronized approaches, a simple list is a sufficient container. There are approaches, in which each thread manages two private lists to store the vertex frontiers and uses additional lists as buffers for communication [18] [19]. This approach deploys a static one dimensional partitioning of the graph's vertices and therefore supports data locality.

IV. ALTERNATIVE TO ATOMIC ACCESSES

Atomic operations in a higher level parallel API for shared memory systems as mutual exclusion, atomic update, locks, compare-and-swap etc. are usually mapped on shared memory systems to atomic instructions that the underlying processor architecture provides. These atomic instructions are by itself rather costly if no contention exists. But if multiple threads concurrently access a shared state with such instructions, the cost *per operation* increases significantly. Fig. 2 shows the cost for one lock/unlock-operation (`omp_set_lock/omp_unset_lock`) in OpenMP on a shared memory system dependend on the number of processor cores utilised. In this test, p processors do in a loop n lock/unlock-operation with an empty function call between that. The test was executed on a large 64 core AMD based system. Other systems show a similar behaviour.

Looking at the formulation of the parallel BFS algorithm in Fig.1, an atomic CAS-Operation is used in line 11 to check whether the child vertex w is unvisited ($d[w] = \infty$), and if so, replace the depth-value of w with the depth value of the current vertex v incremented by one. And if the neighbour vertex w was unvisited, additionally insert w into the next vertex frontier. The CAS operation guarantees, that every vertex is inserted exactly once into a vertex frontier (detection and mark

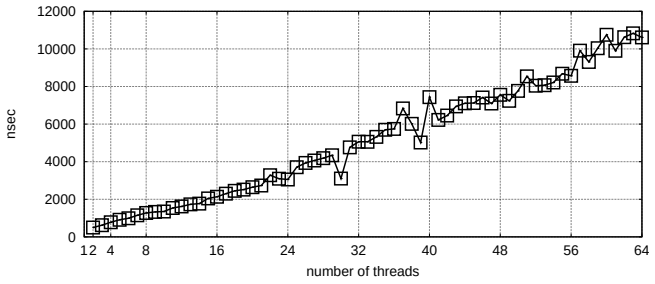


Fig. 2: Cost per lock/unlock on a large AMD-based system.

of visitedness). Without the atomic operation, a race condition exists on $d[w]$. Replacing the critical operation with a non-atomic code results in Fig. 3 (only relevant parts are shown).

```
1: for all neighbours w of v do
2:   if d[w] = ∞ then
3:     d[w] = d[v] + 1
4:     next.insert(w)
5:   end if
6: end for
```

Fig. 3: Parallel non-atomic BFS (relevant part)

The code of interest is in line 2 and 3 that was previously guarded by the CAS-operation. There are two possibilities when executing this code in parallel:

1) Between the read access $d[w]$ in line 2 and the completion of the write acces in line 3 no other thread accesses $d[w]$. In this case (and an appropriate memory model discussed above) there is *no problem* with this version, the vertex w is inserted exactly once in a vertex frontier as before.
2) More than one thread detects for a certain vertex w_x that w_x is unvisited (i.e., $d[w_x] = \infty$) before any of the other threads can change the $d[w_x]$ to some visited value. In this case, the vertex w_x gets inserted twice or even more into the next vertex frontier.

It is important to state that even the second case produces *no wrong results* as *any thread* that detects that $d[w_x]$ is unvisited, writes into $d[w_x]$ in the next step the value $d[v]+1$ that is equal for all threads in one level iteration. Therefore, correctness is guaranteed in our scenario. But, as stated above, in such a case the vertex w_x is inserted twice or even more into the next vertex frontier and due to that, generates more and redundant work in the next level iteration.

Looking at the generated assembler code (and this is more or less invariant of the compiler used), the read access to $d[w]$ in line 2 (i.e., a load instruction) and the write access to $d[w]$ in line 3 (i.e., a store instruction) are nearby instructions in the code sequence. With an assumption, that a thread is not suspended during execution, the time window between the two instructions is therefore rather small (few cycles in practice). This assumption will be mostly true for many real scenarios, e.g., running OpenMP programs on a dedicated system with not more threads than processor cores available.

Another aspect in this discussion is the memory consistency model in use. In a strict memory consistency model, it is guaranteed, that the write operation is visible to other threads immediately after this operation. But todays, all memory consistency models in practical use (e.g., [20] [1]) are rather relaxed and the compiler may buffer the value of $d[w]$ in a register, a processor core may buffer that value in write buffers, or the new value is not propagated between different processors soon etc. This can enlarge the time window for problems substantially even under the assumption made above that a thread is not suspended. A programmer may insert an appropriate flush operation of the used parallel API before line 2 and after line 3 such that all threads / processors are forced to read / write $d[w]$ to / from main memory in the corresponding operation. But dependend on the implementation of such a flush-operation, this could lead to substantial additional overhead as this is done inside an inner loop iteration.

The question we are interested in is now, whether the relaxation using non-atomic modifications to $d[w]$ as given in Fig. 3 (which surely is faster than a CAS-operation) pays off as we might increase the work to be done substantially. The amount of additional work to be done will be influenced generally speaking mainly by:

1) problem time window (influenced by the generated code sequence and implemented consistency model) in relation to the time threads spend in non-critical code
2) the number of threads in use (number of concurrent parties)
3) the problem data influencing access collisions, i.e., in our case the topology of the graph (vertex degrees, shared neighbours)

The larger the time window is that another thread may see the vertex in question as unvisited, and the more threads are participating, and the more vertices have connections to the unvisited vertex, the higher the probability that additional work is generated.

Although we state this here in the context of a parallel BFS algorithm, the discussion is a general discussion on the technique itself and not specific to BFS. We propose to replace costly atomic operations with probably redundant work but with cheaper simple load/store operations without modifying the correctness of the algorithm. The hypothesis is, that especially on large shared memory systems with many concurrent threads this technique pays off.

V. Experimental Setup

In this section, we describe the test setup to systematically compare the two alternatives (atomic accesses vs. redundant work) in the concrete scenario of a parallel BFS. The general algorithmic approach for parallel BFS chosen for this discussion was already given in Fig.1. We optimized this algorithm to work on chunked array based lists where each thread inserts a new vertex into a thread-private chunk. If such a chunk gets filled, the chunk is inserted into a global list. The insertion of a chunk into the global list is done in all algorithm versions with one atomic operation. But the influence of that atomic operation is neglectible.

 ISBN: 978-1-61208-290-5

TABLE I: CHARACTERISTICS FOR USED GRAPHS

graph name	$\|V\|$	$\|E\|$	degree avg.	degree max.	graph depth
RMAT-1M-1G	10^6	10^9	1,000	599,399	8
RMAT-1M-10M	10^6	10^7	10	4,726	16
Streets-Europe	$50,912,018$	$108,109,320$	2.123	13	17,345

In the first version named *atomicBFS*1, every thread uses a CAS operation as described in Fig.1 to detect unvisited vertices and updates them accordingly. This guarantees, that every vertex is inserted exactly once in a vertex frontier. But on the other side, *every* check is done atomically even on vertices that were visited already, even in a previous level iteration.

This last aspect can be optimized easily with a standard optimization technique in prefixing the expensive CAS-operation with a normal read operation followed by the CAS-operation only, if the test was sucessful (i.e., a test-and-test-and-set operation). This technique is also done in the OpenMP reference implementation of the Graph500 benchmark [15] for BFS. We name this version *atomicBFS*2. In this version, all vertices already visited are no longer handled with a CAS operation. We discuss the performance effect of this optimization later.

The third approach (named *nonatomicBFS*) does not use atomic operations for the unvisited-detection, but rather the code shown in Fig. 3. Therefore, a vertex may be inserted more than once in the next vertex frontier. The main difference to the other versions is therefore that the detection of an unvisited vertex and the subsequent update to a visited state is no longer done atomically but rather with simple read/write accesses including the possibility of multiple insertions of a vertex as multiple threads may see a vertex as unvisited concurrently. Further algorithmic optimizations different to that discussed here and a general overview of parallel BFS algorithms can be found in another paper [17]. There is also shown, that there are better but more complex algorithms for the parallel BFS problem. But as we are only interested in this paper in the discussion of atomic operations vs. redundant work, the *relative* comparism of the introduced three versions is sufficient for that.

As we discussed already in section IV, the first factor influencing the probability of multiple insertions is the time window related to the time spent in non-critical code. Although the BFS algorithm has only few instructions between the read and write operation on the critical data, there is not much work to do in the non-critical part. Therefore, BFS is an example for a rather problematic algorithm in this sense.

The second factor influencing the probability of double insertion is the degree of parallelism. We used in our tests different parallel systems. The largest one is a 64 core AMD-6272 Interlagos based system with 128 GB shared memory organised in 4 NUMA nodes, each with 16 cores (1.9 GHz). Another system is a 2-way Intel E5-2670 system with 128 GB main memory and 16-way parallelism (including 2-way Hyperthreading).

The third factor is the probability of a data collision, i.e., two vertices having a common neighbor in the graph. Only unvisited neighbours leed to an atomic operation in version *atomicBFS*1. This factor is mainly influenced in our scenario by the graph topology / degree distribution. We used several large graphs from different application areas. Besides real graphs we used also synthetically generated pseudo-random graphs that guarantee certain topological properties. Due to the limited space in this paper, we will show only results for a street graph (Streets-Europe) and two R-MAT graphs with parameters a, b, c influencing the topology, degree distribution, and clustering properties of the generated graph. See [21] for details on RMAT-graphs and [22] for a general discussion on degree distributions for R-MAT graphs. We used as RMAT parameters in the results shown $a = 0.45, b = 0.25, c = 0.15$. After graph generation, we introduced artificial edges to get a connected graph. Table I shows some important properties of the graphs used.

VI. RESULTS

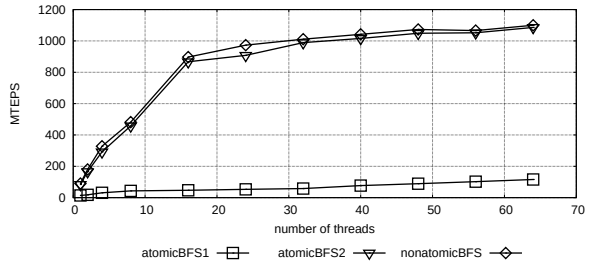
Fig. 4 and Fig. 5 show performance results for the three versions of investigation on the two different parallel systems using different data sets. The performance is given as a rate Million Traversed Edges per Second (MTEPS), a usual measure for BFS performance (the higher, the better). The relative performance degradation Fig.4b and 4c in all versions with higher thread numbers is caused by memory bandwidth restrictions. Details on that can be found in [17].

The unoptimized atomic version *atomicBFS*1 is in all tests slower than the other two versions as with *every* access to $d[w]$ in the relevant code section an atomic operation is executed. The performance difference to the other versions is very high, if many of the atomic operation were done unneccesarily, i.e., a vertex of investigation was visited already before (e.g., Fig. 4a and Fig.5a). For the two atomic versions, most times the optimized second atomic version *atomicBFS*2 is much better due to the prefixed test done with a normal read operation.

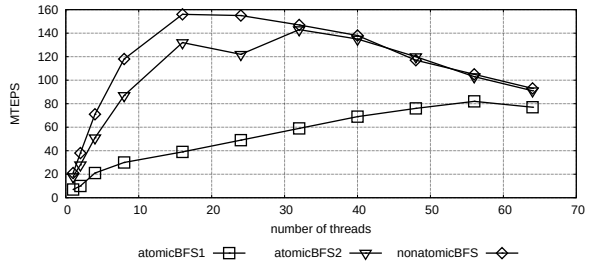
But the best version out of the three is the version *nonatomicBFS* using our proposed technique without any atomic operation in the code section of investigation. The difference to the better atomic version *atomicBFS*2 is rather small if there is a lot of vertex sharing (e.g., vertices have high degrees). In that case, vertices may get visited very often and only the first visit leads to a CAS operation in version *atomicBFS*2 (see again Fig. 4a and Fig.5a). On the other side, the difference between the non-atomic version *nonatomicBFS* and *atomicBFS*2 is quite high, if update operations are done more frequently on vertex visits, as for example in sparse graphs with small vertex degrees (Fig.4b, 4c, 5b, 5c).

To further examine these results, we determined frontier sizes during each level iteration. The *edge frontier size* gives the number of outgoing edges from vertices in the current frontier, i.e., the number of vertex candidates that have to be checked for inclusion into the next frontier. On the other side, the *vertex frontier size* gives the number of unique vertices that get inserted into the next vertex frontier (i.e., the vertex was checked, found unvisited, and then sucessfully inserted). The edge frontier size is therefore the amount of checks to be done (in algorithm version *atomicBFS*1 with a CAS operation, in the other versions by a simple read operation), and the vertex frontier size is the amount of actual insertions into the next frontier (in version *atomicBFS*2 with a CAS, in version *nonatomicBFS* with a simple write). Fig. 6

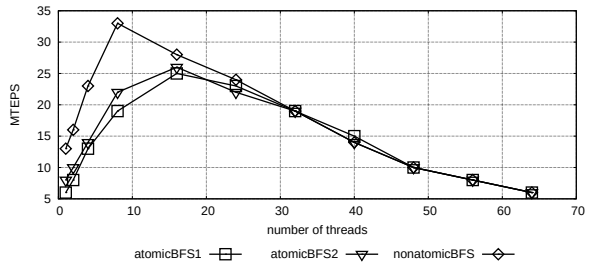
 ISBN: 978-1-61208-290-5



(a) Performance for RMAT-1M-1000M

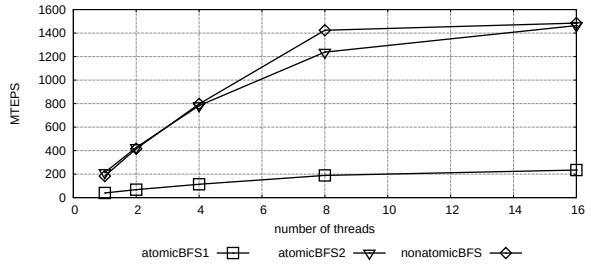


(b) Performance for RMAT-1M-10M

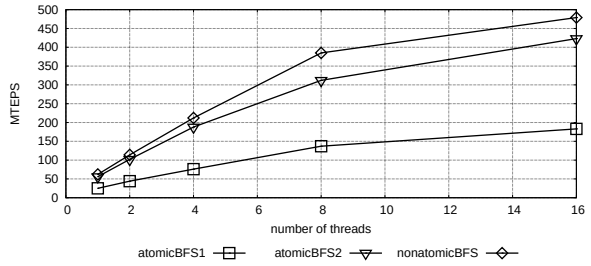


(c) Performance for Streets-Europe

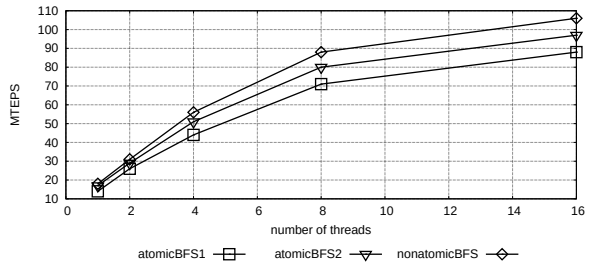
Fig. 4: Performance data on AMD-based system with 64x parallelism.



(a) Performance for RMAT-1M-1000M



(b) Performance for RMAT-1M-10M



(c) Performance for Streets-Europe

Fig. 5: Performance data on Intel-based system with 16x parallelism.

TABLE II: PERCENTAGE OF VERTICES THAT GET INSERTED MULTIPLE TIMES.

number of threads	min. percentage	median	max. percentage
2	0.000012	0.000030	0.000049
4	0.000002	0.000014	0.000026
8	0.000004	0.000013	0.000027
16	0.000002	0.000018	0.000039
24	0.000006	0.000020	0.000037
32	0.000010	0.000022	0.000035
40	0.000010	0.000022	0.000037
48	0.000010	0.000026	0.000035
56	0.000012	0.000027	0.000055
64	0.000016	0.000029	0.000051

shows frontier sizes during each level iteration. Setting this information in relation to the performance numbers, a large difference between edge frontier size and vertex frontier size in a level iteration means that many atomic checks were made in version *atomicBFS*1 that didn't lead to an unvisited neighbor vertex / insert operation. On the other side, if the difference between vertex and edge frontier size is small, the difference between the three versions is less, as the amount of critical operations is rather small compared to all operations executed.

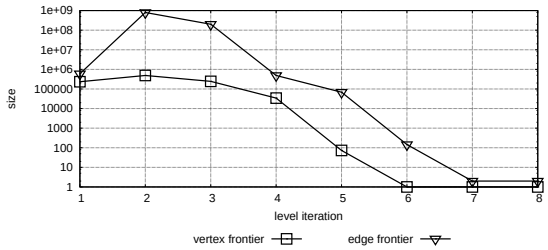
Furtheron, we measured how many vertices get inserted multiple times in version *nonatomicBFS*, i.e., the additional and redundant work that is generated. The factors influencing that were discussed already. We show results only for the largest system and for the street-graph, this is the most problematic test instance where the probability for double insertion is highest. In Tab. II we show the overhead in percentage of vertices inserted more than once, i.e., leading to redundant and more work. As can be seen, the probability increases slightly with more threads, but still this overhead is for our scenario negligible (also in all other tests with different data sets not shown here). Even with 64-fold concurrency, there are very rare situations that lead to multiple insertions. The maximum overhead value is 0.000055 percent or absolutely seen, instead of 50,912,018 vertices to be inserted, with *nonatomicBFS* 50,912,046 vertices were inserted, the difference is 28.

VII. CONCLUSIONS

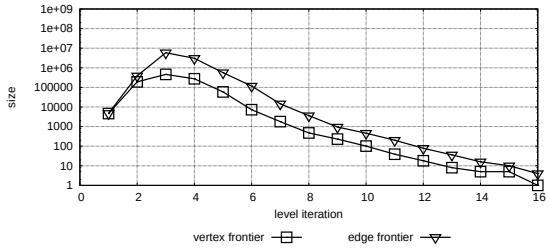
We propose in parallel programs, and within certain scenarios, to replace costly atomic update operations on shared data structures with simple read-write updates. If the correctness of the algorithm is not affected by this change, this leads to an algorithm variant that does not need any atomic operations. This algorithm variant still works correctly, but on the other side, it may generate more and redundant work to be done.

As an example for such a scenario, we used a parallel BFS algorithm where the atomic detection and update of

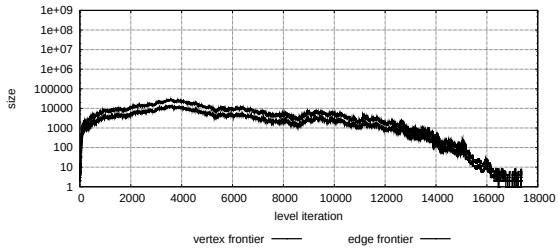
 ISBN: 978-1-61208-290-5



(a) Frontiers for RMAT-1M-1000M



(b) Frontiers for RMAT-1M-10M



(c) Frontiers for Streets-Europe

Fig. 6: Vertex and edge frontier sizes.

univisited neighbour vertices was replaced with simple non-atomic read/write updates. The results show, that for this scenario the non-atomic version has a huge performance improvement in many situations compared to a straightforward implementation with atomic accesses (*atomicBFS*1). And our version has most times a performance improvement of up to 50% compared to an optimized atomic version (*atomicBFS*2) that uses atomic accesses only if necessary. The higher the frequency of atomic operations, the greater the advantage is. Our proposed technique delivers in *all tests* equal or better performance results within the error of measurement than any of the versions with atomic operations.

The upcoming mainstream transactional memory hardware implementations (e.g., Intel Haswell) use a different approach. But similar to our approach, this is an optimistic approach, too, as only the conflict case has to be handled, and not every access. It would be rather interesting to compare these two alternatives with relevant scenarios.

ACKNOWLEDGEMENTS

The system infrastructure was partially funded by an infrastructure grant of the Ministry for Innovation, Science, Research, and Technology of the state North-Rhine-Westphalia. Matthias Makulla did most of the implementation work on several parallel graph algorithms including an initial version of the ones used in this paper.

REFERENCES

[1] "OpenMP application program interface," OpenMP Architecture Review Board, http://www.openmp.org/, 2011, retrieved: 6,2013.

[2] M. Herlihy and N. Shavit, The Art of Multiprocessor Programming. Burlington, MA: Morgan Kaufmann, 2008.

[3] M. Ben-Ari, Principles of Concurrent and Distributed Programming. Harlow: Addison-Wesley, 2006.

[4] V. Agarwal, F. Petrini, D. Pasetto, and D. A. Bader, "Scalable graph exploration on multicore processors," in ACM/IEEE Intl.Conf. for High Performance Computing, Networking, Storage and Analysis, 2010, pp. 1–11.

[5] P. E. McKenney, "Synchronization and scalability in the macho multicore era," http://www2.rdrop.com/∼paulmck/scalability/paper/MachoMulticore.2010.08.09a.pdf, 2010, retrieved: 6,2013.

[6] M. M. Wu, "Asynchronous algorithms for shared memory machines," Ph.D. dissertation, University of Illinois at Urbana-Champaign, 1992.

[7] P. Diniz and M. Rinard, "Synchronization transformations for parallel computing," in Proc. ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL), 1997, pp. 187–200.

[8] D. Novillo, R. C. Unrau, and J. Schaeffer, "Optimizing mutual exclusion synchronization in explicitly parallel programs," in Proc. 5th International Workshop on Languages, Compilers, and Run-Time Systems for Scalable Computers, 2000, pp. 128–142.

[9] M. Desnoyers, P. E. McKenney, A. S. Stern, M. R. Dagenais, and J. Walpole, "User-level implementations of read-copy update," IEEE Transactions on Parallel and Distributed Systems, vol. 23, no. 2, 2012, pp. 375– 382.

[10] D. Dice, "Implementing fast Java monitors with relaxed locks," in Proc. JavaTM Virtual Machine and Technology Symposium, Monterey, 2001, pp. 79–90.

[11] S. Haldar and K. Vidyasankar, "Constructing 1-writer multireader multivalued atomic variables from regular variables," Journal of the ACM, vol. 42, no. 1, 1995, pp. 186–203.

[12] K. Fraser and T. Harris, "Concurrent programming without locks," IEEE Transactions on Computers, vol. 25, no. 2, 2007, pp. 1 – 44.

[13] C. Leiserson and T. Schardl, "A work-efficient parallel breadth-first search algorithm (or how to cope with the nondeterminism of reducers)," in 22nd ACM Symp. on Parallelism in Algorithms and Architectures, 2010, pp. 303–314.

[14] M. Herlihy and J. B. Moss, "Transactional memory: Architectural support for lock-free data structures," in Proc. 20th Intl. Symposium on Computer Architecture, 1993, pp. 289–300.

[15] Graph 500 Comitee, "Graph 500 benchmark suite," http://www.graph500.org/, retrieved: 6, 2013.

[16] D. Bader and K. Madduri, "Snap, small-world network analysis and partitioning: an open-source parallel graph framework for the exploration of large-scale networks," in 22nd IEEE Intl. Symp. on Parallel and Distributed Processing, 2008, pp. 1–12.

[17] R. Berrendorf and M. Makulla, "Parallel breadth first search algorithms for multicore- and multiprocessor systems," in submitted for publication, 2013.

[18] A. Yoo, E. Chow, K. Henderson, W. McLendon, B. Hendrickson, and U. Catalyurek, "A scalable distributed parallel breadth-first search algorithm on BlueGene/L," in ACM/IEEE Conf. on Supercomputing, 2005, pp. 25–44.

[19] Y. Xia and V. Prasanna, "Topologically adaptive parallel breadth-first search on multicore processors," in 21st Intl. Conf. on Parallel and Distributed Computing and Systems, 2009, pp. 1–10.

[20] ISO/IEC 14882:2011 Programming Languages – C++, ISO, Geneva, Switzerland, 2011.

[21] D. Chakrabarti, Y. Zhan, and C. Faloutsos, "R-MAT: A recursive model for graph mining," in SIAM Intl. Conf. on Data Mining, 2004, pp. 442 – 446.

[22] C. Groër, B. D. Sullivan, and S. Poole, "A mathematical analysis of the R-MAT random graph generator," Networks, vol. 58, no. 3, 2011, pp. 159–170.

 ISBN: 978-1-61208-290-5

Proactive Automated Dependable Resource Management in Cloud Environments

Anna Schwanengel and Gerald Kaefer
Siemens AG
Corporate Technologies / Industry Sector
Munich / Nuremberg, Bavaria, Germany
Email: {anna.schwanengel.ext, gerald.kaefer}@siemens.com

Claudia Linnhoff-Popien
Ludwig-Maximilians-University Munich
Institute for Informatics
Munich, Bavaria, Germany
Email: linnhoff@ifi.lmu.de

***Abstract*—Cloud Computing comes along with easy and self-managed resource provisioning and releasing. However, booting and shutting down of instances still means dealing with latencies, administrative efforts and supplementary costs. Considering that, scaling of required resources needs to be well-scheduled, especially, as resources in Clouds are often highly and complexly dependent among each other. The challenge, thereby, is to manage Cloud services with less overhead while fulfilling negotiated SLAs. To automatically provide the exact amount of required instances, our approach enables the detection of resource dependencies and the automated scaling of them without the need for observing the utilization of every single instance. For that reason, we introduce a model addressing resource dependencies and a self-calibration process of the dependency graph used by a regulation method for the dynamic management of dependent resources.**

Keywords*—*Resource Management, Dependencies, Cloud.

I. INTRODUCTION

Although Cloud Computing has set new standards regarding redistribution of virtual machines [1], especially dynamic integration of physical resources, some challenges remain [2], [3]. In particular, the allocation and shut down of these resources as well as their distribution on the same hardware requires a minimum of time [4]. Since a Cloud service does not know in advance when a client plans its usage, an efficient resource planning is not fully automated until now [5] and rule-based mechanisms need to be conducted by the Cloud user [6]. However, in industrial environments, deterministic behaviour is a fundamental requirement and resource availability should be guaranteed for every service, even though they share a pool of physical hardware. Negotiated Service Level Agreements (SLAs) have to be fulfilled and high quality of service should be ensured to satisfy Cloud user interests in time. There is still an enormous need of automated and efficient reaction upon variable resource demands to reduce the amount of precautionary allocated machines, which may then be underutilized most of the time in normal operation causing unnecessary costs.

In this context, resources of Cloud service deployments often show high complex dependencies among each other, as they have a multi-layer structure [7] and avail themselves of other services on the same layer (i.e., composite services). Thereby, a service normally consists of external facing nodes (e.g., Web servers) and internal dependent resources which are required to provide the service. Additionally, resources are shared between the single services, for the purpose of an optimized service-oriented architecture. These dependencies are essential to be considered in service offerings.

Consequently, the question raises how to manage these services while causing less administrative overhead in complex Cloud environments. The goal is to offer a proactive automatic instance management of dependable resources. To achieve this, we enable the automated detection of resource capacity dependencies for supplying the requesting clients with an exact amount of resources required during operation. Then, the scaling process can be done based on the dependency model without having to observe the utilization of each instance.

The paper is organized as follows: Section II gives an overview about research on load management and dependencies in Clouds. The fundamental developed protocol for reservation and feedback based load management through a Service Load Manager (SLM) is pointed out in Section III. The construction of the treated environment and its dependency model is described in Section IV. Section V explains our approach for deducting the capacity demands for dependable resources with the self-calibration and resource regulation methods. Section VI outlines the implementation and results and Section VII concludes and demonstrates future work.

II. RELATED WORK

Load management is important in Clouds and studied by many researchers. The 'SigLM' system, e.g., concentrates on exact resource allocation in shared Cloud environments through fine-grain signatures [8] and Chen et al. use patterns to forecast load (not automated by now), which means trusting historical data and predicting the future [9]. However, without considering dependencies of resources which cover emerging loads on service usage, they waste potential for cost savings.

Brandic [10] wants to support applications according to predefined schedules. While minimizing user interaction with services, she focuses on failure minimization instead of efficiency and performance – we intent to improve the latter. A resource provisioning algorithm for the generation of reservation plans and for the reduction of total provisioning costs is formulated by Chaisiri et al. [11] However, cost factors are here the driving force and again performance issues are disregarded.

Takahashi et al. [7] identify issues emerging under the multi-layered resource environment of Clouds, while concentrating on problems caused by the damage of a single resource

 ISBN: 978-1-61208-290-5

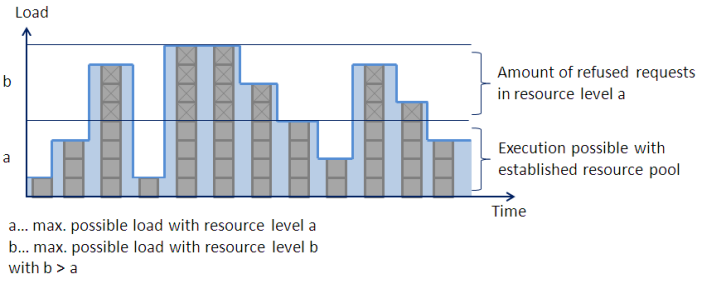


Fig. 1: Load progress of aggregated client requests without delay

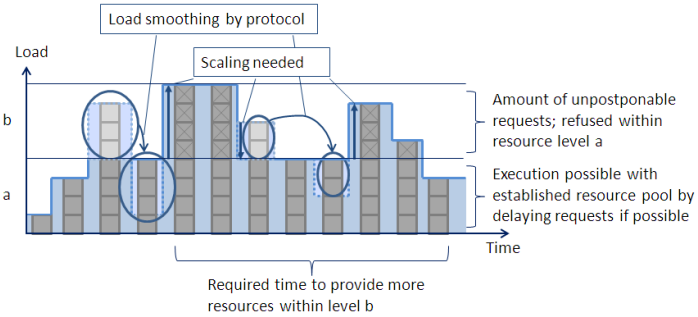


Fig. 2: Load progress with delay of client requests and scaling processes

affecting other resources, which (in-)directly use the faulty one. While creating a dependency graph, as we do, they focus on the aspect of failure tolerance when parts of the system break down. Furthermore, Takahashi et al. build on a very inflexible basis that requires an administrator to monitor all resources and their dependencies. In contrast, we implement active booting and shut down by automatic processes to save these efforts.

The SWAP system, developed by Zheng and Nieh [12], enables automatic dependency detection, too. They use system operation histories to determine resource dependencies among processes and consider them in scheduling. However, their scheduler only has fixed resource pools and scaling processes, which we are concentrating on, are not considered. Also the analyzed algorithms in [13], which are implemented to support optimal provisioning for multiple a priori known tasks, do not consider that the resource pool is changed during operation. In [6] auto-scaling scheduling is proposed, which finishes submitted jobs within specified deadlines considering costs. However, they need to constantly monitor workload changes and, again, newly submitted jobs are ignored in their test bed. Though an automated scheduling architecture managing changes in the Cloud workflow, especially in peak-load situations, is presented by [14], this work lacks the scheduling of tasks regarding their dependencies – unlike we do.

The optimization of scheduling mechanisms have been studied for decades, as in [15]–[17], etc. However, on these approaches, relatively static directed acyclic graphs [18] are assumed, which are given by administrators and elasticity and fast changing environments as common in Cloud Computing are not supported. To sum it up, load management still lacks of fully automated and highly efficient scaling processes, and more studies are needed within this research field.

III. Load Smoothing by the Protocol ReFeLoMaP

In the proposed solution for automatic scaling in Cloud environments, we base on our developed environment already described in [19] and [20], and extend this approach by the possibility to deduct capacity demands for dependable resources. In previous work, we created a technique to dynamically manage the load of Cloud services based on resource reservation and service feedback and a method to influence the behaviour of service usage by the service itself. The implemented light-weight protocol ReFeLoMaP handles the communication between the Cloud services and their clients via a service load manager, which coordinates the actual resource demands and their availability. In the following, we build on this procedure and define a self-calibration method before the process as well as a resource regulation algorithm of the SLM during active operation.

As outlined in [20], the mediating SLM initially aggregates all incoming client requests, and in this way, the load process can be depicted in a graph as shown in Figure 1. In highly distributed systems, client requests which exceed the actually provided resource capacity are normally refused in high load peak times in order to remain operational until additional resources are allocated. To avoid this ineffective manner of operation our protocol ReFeLoMaP delays specific requests about a defined time interval as agreed upon with the corresponding clients. After having delayed these client requests, one can observe less underutilized machines. Respectively, scaling processes are less frequent needed in order to cover all clients' demands. Consequently, we are enabled to smooth load peaks, which leads to better instance utilization and less frequent scaling demands, as illustrated in Figure 2.

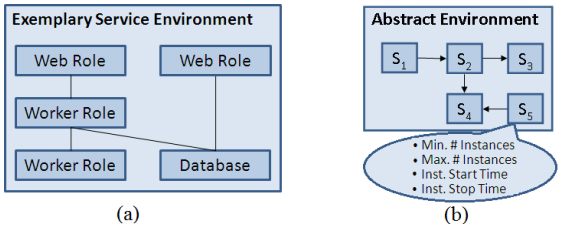
The dependencies among every single resource need to be well known in order to be capable of deciding automatically which and how many virtual instances can be booted or shut down on these resources. More precisely, the ratio of running machines on each specific layer must be computable. That way, it is possible to scale the instances of a service with its respective dependable resources at once without having to evaluate the utilization of each resource in case of load variations and unpredicted load peaks.

IV. Construction of Service Environment and its Dependency Model

In order to get a better understanding of the typical multi-layered composite service architecture of Clouds, imagine a service environment consisting of several Web and Worker instances as well as databases for storing persistent information. Instances of the same type can be clustered to groups - called roles. Such a system may look as depicted in Figure 3 (a), wherein a shared database and a composite service are located - the latter comprising one Web role, two Worker roles and exactly that database. More abstractly, in this example, the service environment exists of five different services S_1 to S_5.

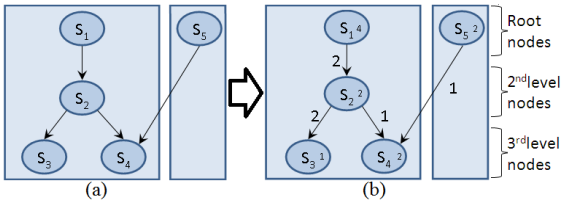
At first, we define which services are related to which others in order to build up a dependency graph for scaling processes. For that purpose, every service indicates its relationships to others on the initial registration at the SLM. The SLM

 ISBN: 978-1-61208-290-5



(a) (b)

Fig. 3: Service environment (a) and its abstraction stored at the SLM (b)



(a) (b)

Fig. 4: DAG of the service environment at the SLM

stores these connections and sets up attributes about the passed on minimal and maximal amount of instances per each scalable resource (Min./Max. # Instances) and the time, a service needs for starting and stopping of new instances (Inst. Start/Stop Time). With reference to our example, on registration at the SLM, the Web role (represented by service S_1) indicates a dependency to the Worker role (service S_2), S_2 a relation to another Worker role (service S_3) as well as to the shared database (service S_4). Another Web role (service S_5) states a direct dependency to this database S_4, too (see Fig. 3 (b)).

In order to organize the dependencies of all listed services at the SLM, we start up from common Directed Acyclic Graphs (DAG) $G = (V, E)$, where V is a set of v_i nodes and E is a set of e_i directed edges. In [18], DAG-nodes represent tasks and the weight w of a node n_i is called the computation cost $w(n_i)$. An edge is represented by (n_i, n_j) and its weight (communication cost) is given by $c(n_i, n_j) = \frac{w(n_i)}{w(n_j)}$. Based on this, we apply the traditional DAG and modify its usage. In our implementation the nodes are the different Web, Worker or database roles registered at the SLM and their weight is the number of running instances of a role. In our example, we have a node set of $V = \{S_1, ..., S_5\}$. The weight of an edge represents the ratio between the two associated roles. If the Web Role S_1 has, e.g., four running instances and the underlying Worker Role S_2 needs two instances, their ratio of running instances is 4:2 with leads to a ratio of 2.

For the automated graph generation, shared resources must not be influenced by other related ones, because the instances can not be used simultaneously by different roles at the same time. Therefore, in the detection process, resource S_5 should be inactive when defining the graph of S_1, S_2, S_3 and S_4. Vice versa the services S_1 to S_4 must be idle on the graph creation of S_5, because, with using the same instances, the different sequence threads can not be parallelized. When defining the capacity demands for the resources, we then can guarantee always the same workload and no sum of load at each resource.

V. Deduction of Capacity Demands for Dependable Resources

Having defined the essential fundamentals of the distributed system, we got a starting point for the further solution. Our following approach consists of two procedures to automate the scaling of dependable resources in service provisioning of Cloud Computing systems. First, we implemented a self-calibration process at the SLM before productive operation. This technique for generating the dependency graph as well as setting up the initial amount of resources on each system layer is explained in Subsection V-A. Secondly, we describe the resource regulation method of the SLM during the active operation in Subsection V-B.

A. Self-calibration before Operation at the SLM

In order to be capable of scaling the whole composite service without monitoring every single instance utilization, the SLM sets up the dependencies between the resource-instances on the different layers in a DAG (see Figure 4(a)). The ratio between the instances of dependable resources can be either defined manually by an administrator or, as in our approach, automatically with boundary conditions. With this management by the generated dependency graph, organizational efforts can be minimized, as we do not have to constantly observe each resource utilization by an administrator. For automatic ratio detection of the instance amount per each dependable resource, the SLM specifies a target value for each resource-instance regarding, e.g., the average CPU or RAM utilization. For instance, it can be determined that the average CPU load must not exceed 50%. Although, in real scenarios, the server utilization in a datacenter is estimated to range only from 5% to 20% [21], [22], we are able to offer a higher utilization because of better knowledge of future resource demands based on our developed protocol ReFeLoMaP [20].

In a second step, the SLM defines a default value in the configuration file for each service and boots according to that value the amount of resources. The minimal amount of instances of service S_1 (Min. # Instances S_1) is, e.g., two virtual machines: $w_{min}(S_1) = 2$. Then the regulation process starts: a fictional load is generated on the first node of the graph (root node), which is the external facing resource of the service, and it passes, thereby, the depending load to behind nodes. As stated before, the load of shared resources is only generated at the first node in order to not falsify the measurement, and then, passed on by the first service node. So, we can exclude accumulated load values and guarantee reproducible test scores. This implies that other services are not active during the automated detection of the amount of needed instances for achieving the target value.

Afterwards, the SLM allocates instances on the according resource (role) until the previously set target value is reached and stores the corresponding amount of running instances ($w_{act}(n_i)$) in the nodes as its weight ($w(n_i)$). Analogous to that, the SLM proceeds for all depending resources on lower layers and adjusts the instance amount for each of them. In our implemented test scenario (see section VI), the Web role S_1 holds four instances and the underlying Worker role S_2 needs two instances. This Worker role requires one active instance at the lower Worker role S_3 and two running machines at the database service S_4 in order to achieve the corresponding target

 ISBN: 978-1-61208-290-5

values regarding CPU utilization and storage occupancy. In a second iteration, the required amount of instances in Web role S_5 can be set on two with an according process. By means of the list of the instance amount of every single layer, the role ratios can be defined. In our case, Web role S_1 holds four instances ($w(S_1) = 4$) and the underlying Worker role required two ($w(S_2) = 2$). Consequently, their ratio is 2, which is also the weight of the corresponding edge:

$$c(S_1, S_2) = \frac{w(S_1)}{w(S_2)} = \frac{4}{2} = 2 \quad (1)$$

This Worker role has a weight of 2, similar to the other Worker role S_3 ($c(S_2, S_3) = \frac{2}{1} = 2$) and a ratio of 1 to the database ($c(S_2, S_4) = \frac{2}{2} = 1$). The ratio of the Web role S_5 to the shared database S_4 is 1, too ($c(S_5, S_4) = \frac{2}{2} = 1$). As shown in Figure 4(b), the ratio and the amount of instances are inscribed at the edges and in the nodes of the dependency graph. Following this, the basic dependency graph is completed with concrete values by the SLM after the registration period.

After this completion, the regulation process is stopped and the required instances per each resource can be scaled during operation based on the dependency graph. The self-calibration procedure can be triggered periodically in order to correct the specified dependency values as appropriate.

B. Resource Regulation Process of the SLM

With this dependency graph, the SLM is able to regulate the resource amount during operation, described as follows. First, all composite services (meaning all depending resources) are reduced on one virtual resource. As a result, the corresponding limits regarding the minimum and maximum of instance amount and their booting and shut down time are well known for all composite services. In the case, there are shared resources (as database S_4), the maximum aggregated limit between the dependable resources is additionally defined.

In every single regulation step, the SLM:

1) accumulates all of the client requests on each virtual provided service, then,
2) controls the limit checks on basis of the external facing node accessed by the client and
3) checks the aggregated limits of the dependable services in case of shared resources.

The SLM modulates the role instance counts according to the relations between the roles and their defined weights of nodes and edges, as long as the aggregated instance counts stay within defined limits. Thereby, the SLM does not deceed the minimal amount of resource-instances predefined by the services themselves before their actual execution – nor exceed their corresponding maximum. For instance, we define a minimal amount of least two and maximal ten instances at resource S_1. Then S_1 will always run with at least two instances, even though, one would be sufficient to cover the actual load. That way, it is possible to react on unexpected load peaks if needed. If anon in S_2 we have at least one and maximal four instances, the maximum of four instances in S_2 is not extended, even though S_1 is scaled up to five and higher instance amounts.

In the case, aggregated limits of shared resources would be exceeded, a load release occurs according to the determined service priority defined by the protocol ReFeLoMaP. On equal priority the sharing is defrayed in equal proportions. Thereby, we can additionally focus on maximization of throughput, meaning the service delays specific client requests about a defined interval. This is possible even then the client is running in a higher priority class of the service level agreement as long as the client allows its procrastination by the manipulation of the service load manger. Thereby, also lower prioritized client requests need not to be dropped. For further information about the concrete implementation and the benefits of the protocol ReFeLoMaP, see our paper [19].

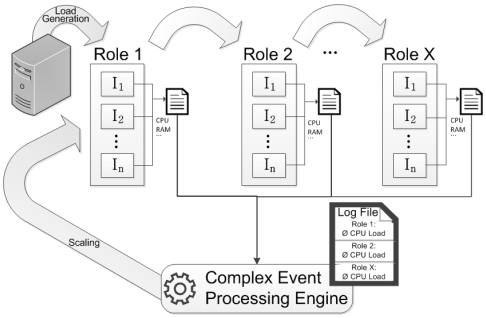


Fig. 5: Implementation of the Automated Scaling Process

If no aggregated limits are exceeded but particular service limits of virtual external facing nodes, the client load management feedback is generated on basis of this maximum load in order to distribute the load in a way that load dropping through request refusing can be avoided.

VI. Implementation and Results

Having defined the dependency graph and regulation method, the SLM is now able to regard dependable resources during the instance management process. Figure 5 provides an implementation overview of this control algorithm. As shown, an external server, which runs the SLM, generates a fictional load on $Role\ 1$. $Role\ 1$ now boots new instances ($I_1, ..., I_n$) until the target value of 50% CPU and RAM workload is achieved and stores the number of running instances with their average utilization in an internal log file. The load is passed on to all subsequent roles ($Role\ 2, ..., Role\ X$) and they proceed accordingly. After having completed the log files, they are transferred to a complex event processing engine, which analyses the data streams from the resources about the happening events, consolidates all values and passes it on to the SLM. The SLM integrates this data in the dependency graph and starts scaling corresponding to this. If an instance brakes down, it is replaced by a self-healing routine of the roles.

In a first simulation for testing the time constraints about the generation of a corresponding dependency graph, we could make the following observations. We successively simulated the registration of 5, 10, 20, 30 and 50 services at the SLM. All of them indicated random dependencies to each other. As shown in Figure 6, we took the measurements after which time the complex event processing engine announces the completion of the dependency graph generation. With an amount of 5 services it took 3 seconds until all dependencies were stored in the graph with their corresponding weights regarding instance

 ISBN: 978-1-61208-290-5

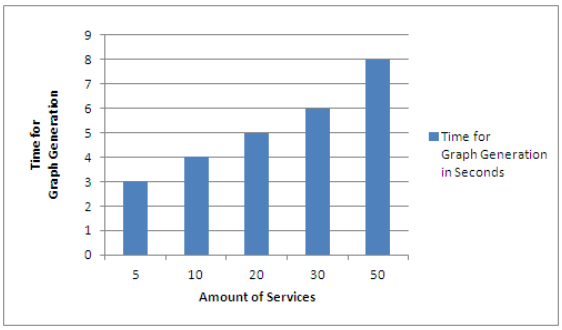


Fig. 6: Simulation of the Graph Generation Process

amount and ratios to each other. With 10 services, the time-demand linearly rises to 4 seconds, and with 20 services, to 5 seconds. On the registration of 30 services at the SLM, we measured 6 seconds and with 50 services we had 8 seconds for the graph creation. On the first glance, these values increase more and more. However, we assume that after the initial graph generation the time demand for dependency detection even with more services will not increase too extensively as the values do only rise sparsely. Furthermore, since we are concentrating on services with no real-time requirements, we can hazard these consequences and accept this delay to benefit from the more efficient service provisioning as a whole.

In a further realized scenario, we simulated an rising load requirement at the external facing node of the clients (i.e., Web service S_1), which results in a new demand of two additional instances on the resource of S_1 ($w_{act}(S_1) = 6$). Consequently, an upregulation of the related services S_2, S_3 and S_4 (Worker roles and database service) is needed, too. Following the regulation by the dependency graph, the SLM initiates the booting of an additional instance in the Worker role S_2 on the second layer, because the initial defined weight of their relation is 2 (see Formula 1) and the weight with the additional two instances on service S_1 is now 3:

$$c_{act}(S_1, S_2) = \frac{w_{act}(S_1)}{w_{act}(S_2)} = \frac{6}{2} = 3 \neq 2 = c(S_1, S_2) \quad (2)$$

That means the actual running virtual machines of the Worker role S_2 need support by a further resource-instance in order to achieve the given weight of 2 ($w(S_2) = 3$).

Since the load is passed on to the lower lying database service S_4 also in this resource a new instance is started in order to fulfill the determined weights of the dependency graph ($w(S_4) = 3$). Although the load is also transferred to the Worker role of layer three (S_3), here no additional instances are required: with three instances in S_2 and one already running instance in S_3 the ratio is yet sufficient because its weight is smaller then four (see Formula 3), on which an up-scaling of the resource-instances would be necessary ($w_{act}(S_3) = w(S_3) = 1$).

$$c_{act}(S_2, S_3) = \frac{w_{act}(S_2)}{w_{act}(S_3)} = \frac{3}{1} = 3 < 4 \quad (3)$$

With a second scenario, we covered the case that S_5 is the most stressed service in the system. The high load generated on service S_5 leads to an up-scaling of the database service S_4 through S_5 and S_1 according to the process described before. After a specific time, the internal maximal limit of instances is reached. As a consequence, the maximal amount of instances in the back-end reduces also directly the maximal instance amount of the front-end, and additional instances in S_1 as well as in S_5 cannot be supported by the fully exhausted resource pool of S_4. This situation is recognized by the complex event processing engine with the aggregated information of the log files from each resource and a corresponding warning is send to the SLM in order to take countermeasures. The SLM is now able to interrupt on its higher management level and can counteract an inefficient service provisioning which only supports one part of the composite service while neglecting the majority of the entire resource environment. For that reason, it throttles service S_5 in order to keep the whole composite service functional and reliable.

So, a predictive detection of resource bottlenecks can be conducted. With the aid of the defined resource limit values, the SLM is able to proactively respond to congestions by delaying client requests in times of overload. This prescient bottleneck identification is also possible with shared resources by using the computation of the accumulated resource loads of the composite service. On this basis the SLM can decide which service is postponed according to the specified service priority when exceeding the limit of the shared resource.

By means of the anticipatory recognition of the composite resource limitations caused by dependencies, it is possible to make an assumption of the limits for the virtual external facing node directly accessed by the client. In our example, at its registration, the external facing node S_1 in the root level states that it needs at least two and maximal ten instances ($w_{min}(S_1) = 2$ and $w_{max}(S_1) = 10$). This resource S_2 has anon at least one and maximal four instances ($w_{min}(S_2) = 1$ and $w_{max}(S_2) = 4$). As calculated before, the dependency ratio of S_1 to the hidden resource S_2 is 6:3 (proved by Formula 2). So, with the defined dependency ratio $c(S_1, S_2) = 2$ (see Formula 1), S_1 is allowed to double the instance amount of S_2 as long as it does not exceed its own maximum. Following this reasoning, we can conduct these minimum evaluations:

$$min\{w_{min}(S_1), [2 * w_{min}(S_2)]\} = min\{2, [2 * 1]\} = 2 \quad (4)$$

$$min\{w_{max}(S_1), [2 * w_{max}(S_2)]\} = min\{10, [2 * 4]\} = 8 \quad (5)$$

Consequently, S_1 is restricted in its instance amount by its dependency to service S_2 which has less capacity. So, the virtual resource S_1 has the actual instance limit range of minimal two (see Formula 4) and maximal eight running instances (see Formula 5). Thereby, it is recognizable that an utilization of the technically possible amount of ten instances in S_1 would not create added value.

Another resulting benefit of our approach is the possibility to make estimations, until when the additional capacity is available at the highest level on booting new instances. How long the entire starting time interval is, can be calculated already before based on a maximum evaluation along the dependency graph. In our scenario, booting new Web role instances in S_1 and S_5 needs a time interval of three minutes. An extra Worker role instance in S_2 and S_3 is available after five minutes each and the start of an additional database instance takes nine minutes for installing the base image and the particular software of the tenants. Consequently, in our scenario, it requires a start time consideration of nine minutes for the up-scaling of the whole service, which is the maximum

 ISBN: 978-1-61208-290-5

of these parallel booting instances. Thereby, we are able to give estimations about the time point when the required instances are available earliest and the entire composite service is highly functional, again. On the other hand, it is calculable how long the release of redundant instances takes until unnecessary expense decreases.

In summary, we could observe an enhanced pro-active reaction of system resources upon proactively derived load demands. This is achieved by deriving load for dependable hidden resources from front-end resources instead of following an approach where roles manage their instance counts on local utilization monitoring. Hidden resources can be adjusted directly based on the metrics of the front-end nodes (dependency root nodes) and the dependency graph. This results in a significant improvement of the overall system load reaction.

VII. Conclusion

Load management is a driving force for comprehensive research in the area of Cloud Computing. Although this field already has its roots in basic Utility Computing and ranges from High Performance Computing (HPC) over Grid Computing to the today's era of Cloud Computing, there are still unsolved problems showing space for improvements regarding cost savings and resource efficiency. In Cloud Computing environments, providers offer a theoretically unlimited pool of resources, which Cloud services can benefit from. Thereby, a lot of heterogeneous workloads emerge and one has to react on enormous load variations in time in order to comply with the SLAs concluded with the Cloud clients. For this reason, Cloud systems provide elasticity meaning an easy booting and release of instances while relaxing strong SLAs.

On closer examination, in Clouds sophisticated and intricate dependencies among the involved resources can be observed while offering specific services both on its one and in its entirety as a composition. This is because of the multi-layer structure of Cloud services and leads to considerable efforts regarding administration, costs and time. Within this approach, we offer a management of Cloud services with less administrative overhead within complex Cloud environments. Thereby, we want to offer a proactive automatic instance management of dependable resources. For this purpose, we detect resource dependencies automatically to supply requesting clients the exact amount of required resources. The corresponding scaling process is conducted on basis of the dependency model without the need for observing each instance utilization.

While building on the previously developed protocol for reservation and feedback based load management through a SLM, we constructed a Cloud service environment and described its dependency model. Our approach deducts the capacity demands for dependable resources, while introducing a self-calibration mechanism before the standard operation and a resource regulation procedure by the SLM during operation. We proof the concept by an implementation and simulation. In future, we aim to expand the described load management to offer a functional entity for adding and releasing instances while guaranteeing low costs and SLA compliance by reliable service provisioning. Furthermore, we will prove our approach by additional evaluation with a practical use in Cloud Computing environments. Afterwards, we will compare our results with cloud scaling algorithms based on utilization values.

References

[1] M. Assunção, A. Di Costanzo, and R. Buyya, "Evaluating the cost-benefit of using cloud computing to extend the capacity of clusters," in *18th ACM Int'l Symposium on High Performance Distributed Computing*, 2009, pp. 141–150.

[2] M. Armbrust and et al., "Above the clouds: A berkeley view of cloud computing," University Berkley, USA, pp. 141–150, 2009.

[3] M. Armbrust, et al., "A view of cloud computing," *Commun. ACM*, vol. 53, pp. 50–58, 2010.

[4] M. Mao and M. Humphrey, "A performance study on the VM startup time in the cloud," in *5th Int'l Conference on Cloud Computing*. IEEE, 2012, pp. 423–430.

[5] K. Alam, E. Keresteci, B. Nene, and T. Swanson, "Multi-tenant applications on windows azure: Dokumentation," http://cloudninja.codeplex.com/releases/view/65798, 2011, last access 20.6.2013.

[6] M. Mao and M. Humphrey, "Auto-scaling to minimize cost and meet application deadlines in cloud workflows," in *Conference on High Performance Computing, Networking, Storage and Analysis*. IEEE, 2011, pp. 1–12.

[7] T. Takahashi, Y. Kadobayashi, and H. Fujiwara, "Ontological approach toward cybersecurity in cloud computing," in *3rd Int'l Conference on Security of Information and Networks*. ACM, 2010, pp. 100–109.

[8] Z. Gong, P. Ramaswamy, X. Gu, and X. Ma, "Siglm: Signature-driven load management for cloud computing infrastructures," in *17th Int'l Workshop on Quality of Service*. IEEE, 2009, pp. 1–9.

[9] J. Chen, W. Li, A. Lau, J. Cao, and K. Wang, "Automated load curve data cleansing in power systems," *Transactions on Smart Grid*, vol. 1, no. 2, pp. 213–221, 2010.

[10] I. Brandic, "Towards self-manageable cloud services," in *33rd Software and Applications Conference*, vol. 2. IEEE, 2009, pp. 128–133.

[11] S. Chaisiri, B.-S. Lee, and D. Niyato, "Optimization of resource provisioning cost in cloud computing," *Transactions on Services Computing*, vol. 5, no. 2, pp. 164–177, 2012.

[12] H. Zheng and J. Nieh, "Swap: A scheduler with automatic process dependency detection," in *1st Symposium on Networked Systems Design and Implementation*. USENIX Association, 2004, pp. 145–158.

[13] M. Malawski, G. Juve, E. Deelman, and J. Nabrzyski, "Cost-and deadline-constrained provisioning for scientific workflow ensembles in iaas clouds," in *Conference on High Performance Computing, Networking, Storage and Analysis*. IEEE Computer Society, 2012, pp. 1–11.

[14] T. Dornemann, E. Juhnke, and B. Freisleben, "On-demand resource provisioning for BPEL workflows using amazon's elastic compute cloud," in *9th Inte'l Symposium on Cluster Computing and the Grid*. IEEE, 2009, pp. 140–147.

[15] D. C. Steere, A. Goel, J. Gruenberg, D. McNamee, C. Pu, and J. Walpole, "A feedback-driven proportion allocator for real-rate scheduling," in *3rd Symposium on Operating Systems Design and Implementation*. USENIX Association, 1999, pp. 145–158.

[16] F. Dong and S. Akl, "Scheduling algorithms for grid computing: State of the art and open problems," School of Computing, Queen's University, Kingston, Ontario, Tech. Rep., 2006.

[17] M. Aggarwal, R. Kent, and A. Ngom, "Genetic algorithm based scheduler for computational grids," in *19th Int'l Symposium on High Performance Computing Systems and Applications*, 2005, pp. 209 – 215.

[18] Y.-K. Kwok and I. Ahmad, "Static scheduling algorithms for allocating directed task graphs to multiprocessors," *ACM Comput. Surv.*, vol. 31, no. 4, pp. 406–471, Dec. 1999.

[19] A. Schwanengel and G. Kaefer, "Light-weight load management protocol based on reservation and feedback loops," in *23rd Int'l Conference on Parallel and Distributed Computing and Systems*. ActaPress, 2011, pp. 165–172.

[20] A. Schwanengel, G. Kaefer, and C. Linnhoff-Popien, "Improved throughput and response times by a light-weight load management protocol," *Journal of Parallel and Cloud Computing*, vol. 1, pp. 1–9, 2012.

[21] K. Ragan, "The cloud wars: $100+ billion at stake," Tech. Report, Merrill Lynch, 2008.

[22] L. Siegele, "Let it rise: A special report on corporate it," *The Economist*, vol. 389, pp. 3–14, 2008.

 ISBN: 978-1-61208-290-5

How to Run Scientific Applications with DIRAC in Federated Hybrid Clouds

Víctor Méndez Muñoz
LHC Tier-1 Computing Production.
Port d'Informació Científica (PIC).
Universitat Autònoma de
Barcelona (UAB).
Bellaterra, Spain.
Email: vmendez@pic.es

Adrian Casajús Ramo and
Ricardo Graciani Diaz
Department of Structure and
Constituents of Matter.
Universitat de Barcelona (UB).
Barcelona, Spain.

Víctor Fernández Albor
Particle Physics Department.
Universidade de Santiago de
Compostela (USC).
Santiago de Compostela, Spain.

***Abstract*---Nowadays, the eScience big issue in Cloud Computing is how to leverage on-demand computing in scientific research. For this purpose, the specifics requirements of the complex scientific applications have been addressed with DIRAC, which has the motto *the interware*, because it is a proven scientific community solution, currently providing transparent access and interoperability between different distributed infrastructures, such as European Grid Infrastructure (EGI), Open Science Grid (OSG), computing clusters, standalone hosts, and cloud infrastructures. In this context, Federated Hybrid Clouds are emerging as a model of coordinated service access and delivery to multiple Infrastructure as a Service (IaaS) providers. The term hybrid comes from the integration of community clouds and commercial clouds in a federated manner, which also requires the use of additional services, such as federated authentication, accounting or monitoring. This paper explains how DIRAC is providing Software as a Service (SaaS) for generic scientific computational purposes. The cloud extension of DIRAC (VMDIRAC) is used to instantiate, monitor and manage Virtual Machines (VMs) in multiple IaaS aggregations of Amazon EC2, OpenNebula, OpenStack and CloudStack. Furthermore, DIRAC and VMDIRAC extension can provide SaaS of any scientific application through a contextualization management of few *golden* VM images, which automates the necessary context to run transparently in multiple IaaS providers, as well as the required software and tools for any eScience application.**

Keywords*---*Cloud Computing; Federated Hybrid Cloud; on-demand Cloud Computing models

I. INTRODUCTION

VMDIRAC is the chosen tool in Cloud Computing matters for the scientific computing of LHCb[1] and Belle [2] in high energy physics (HEP) and the different Virtual Organizations (VOs) of the France Grilles[3] using the DIRAC portal, with an important community of life science. There is work in progress for the adoption of a DIRAC portal including the VMDIRAC extension for Federated Clouds in the context of National Grid Initiatives (NGIs) of the European Grid Infrastructure (EGI), as well as DIRAC portals of some scientific communities, like BES[4] and CTA [5] in astrophysics, or ILC[6] (HEP).

The proposed solution represents a big step forward in terms of scope. Any adopter is able to target cloud computing resources transparently, while the storage is supported by third-party solutions. Moreover, this allows the users to take advantage of the virtualization assets, mainly the VM encapsulation opening an opportunity window in the multi-core running [7], to exploits the latest many cores hardware without dependency on the platform, which becomes user specific. Additional opportunities for user requirements in High Performance Computing (HPC) are also addressable by HPC Cloud providers [8], [9], [10].

The term VMDIRAC is used for the specific features of the federated cloud extension, while the term DIRAC is used for the general features or components.

This paper is focused on the IaaS provider and user communities requirements to deploy Federated Hybrid Clouds with DIRAC. It is organized as follows. Section 2 defines how to aggregate an IaaS provider to a DIRAC Federated Hybrid Cloud. Section 3 is focused on the DIRAC setup to run a generic eScience application using the available IaaS providers. Section 4 describes the Web interface for cloud management. The paper finishes with a conclusion section.

II. HOW TO AGGREGATE AN IAAS PROVIDER TO DIRAC

The first step is to have a DIRAC portal with a VMDIRAC extension installed, or request the VMDIRAC installation to the portal administrators. The installation, configuration and operational maintenance of a DIRAC portal is not a trivial matter, so the easy way would be to ask to your NGI for this purpose. Currently, some NGIs have their own DIRAC portals: France Grilles DIRAC portal[11] and also Ibergrid portal[12], which is the Spanish and Portuguese common infrastructure. Other NGIs are considering to deploy they own DIRAC portal, therefore NGI would be the first place to ask for support. For medium and big eScience communities interested in having a DIRAC portal, the official DIRAC webpage[13] may provide instructions for further support.

Once there is a DIRAC portal with a VMDIRAC extension installed, there are few IaaS provider requirements. This section describes the minimal requirements to deploy a Federated Hybrid Cloud using DIRAC, then, the supported IaaS cloud managers and some basic specifications and recommendations to the IaaS providers.

A. MINIMAL REQUIREMENTS TO DEPLOY A FEDERATED HYBRID CLOUD WITH DIRAC

Federated hybrid cloud computing model [14] is considering commercial and community cloud end-points as resources. An overall federated cloud model is including federated services which are necessary in a scientific community. Such federated services definition is usually related with Metadata

 ISBN: 978-1-61208-290-5

repository of images, Information System, Accounting and Monitoring, as well as third-party services, which are offshore of the Federated Hybrid Cloud infrastructure, for example authentication and authorization or external storage.

To facilitate the aggregation of IaaS providers in a federated manner, VMDIRAC is able to deal with minimal requirements that do not require external services, but include manually the necessary information in the DIRAC Configuration Server. This is a compromise solution to allow a fast deployment of the Federated Hybrid Cloud having only the IaaS providers end-points, then VMDIRAC is able to extend and automate specific implementations of such external federated services.

The minimal requirements for a Federated Hybrid Cloud can be deployed in DIRAC as follows:

- Including a metadata repository of images, with the necessary contextualization. Any third-party automated image distribution can be used or manually uploaded to the different IaaS providers, the corresponding metadata of these images is described in the DIRAC Configuration Server.
- The IaaS providers information about the end-points is defined in the DIRAC Configuration Server. This information is usually published in the Information System.
- VM Monitoring for the eScience community users by the VM Browsing of the Web interface, to monitor VM history logs and plots related with transfers, jobs and CPU loads. See Fig. 1.
- External IaaS provider monitoring of the VM running on their site can be included using VM contextualization to deploy the monitoring client. For example, monitoring and notification based on Nagios alarms[15] or VM statistics monitoring with Ganglia[16]. This is an IaaS provider requirement, which is optional to VMDIRAC.

The mentioned third-party services are transparent from DIRAC point of view, because the interaction with them is not directly assumed by VMDIRAC. Thus, scientific applications can use external storage resources. The current approaches are using Grid and Cloud Storage, such as standards gridftp[17] and Cloud Data Management Interface (CDMI[18]), a pay-per-use storage like Amazon S3[19] or external storage resource interfaces [20] [21]. Some of these storage resources can be supported as part of the IaaS resources. Authorization and authentication methods associated to different user access to the end-points, requires from DIRAC a minor support because Configuration Server has the information related to the user, X509 proxy or other user identifications. At the same time, VMDIRAC is not interacting with the third-party authentication service but with the IaaS provider, which transparently supports the authentication and authorization. In general terms, the IaaS *auth* is managed on VO basis, the VM are created and owned by the VO, not by a particular user. Once the VM is created DIRAC takes control supporting DIRAC user policies. VM can run multiple jobs and each one is corresponding to a particular DIRAC user. This job *auth* management is fully supported by DIRAC, which eventually may contact external services for proxy or token authorization and it is decoupled from the IaaS *auth* management.

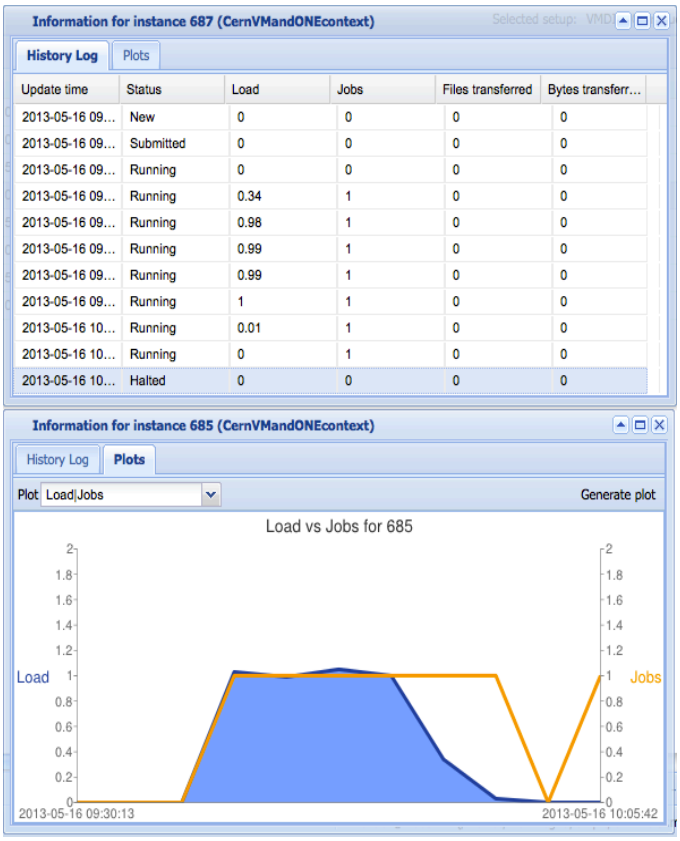


Fig. 1: DIRAC VM Monitoring for the eScience community

B. SUPPORTING IAAS PROVIDERS IN DIRAC

Current VMDIRAC release supports the following cloud managers APIs for commercial and private clouds:

- Amazon EC2
- OpenNebula OCCI 0.8
- OpenStack Nova 1.1
- CloudStack 2

It is necessary to have at least one of such end-point deployed in the cloud manager server. The VM needs to have out-bound connectivity to the VMDIRAC server.

From the scaling test in some IaaS providers [22] [23] [24], there are some known lessons that should be considered:

- To scale up in the IaaS site it is necessary to have a snap-shot image management, just to accelerate the instance creation in the host hypervisors. This has been particularly tested with OpenNebula, which can use *qcow2* and *NFS* for the image distribution and creation in the host hypervisors.
- The opportunistic use of the hypervisor memory among the VMs it is highly discouraged. This has been particularly tested with KVM hypervisor. The result

 ISBN: 978-1-61208-290-5

shows the average performances are not improved, while the dispersion on the performances is increased.

- The hypervisor kernel flags should be compatible with the software to run. In particular, KVM hypervisor is using *lib viewer*, which is working transparently on *Intel* platform for the wide range of scientific software tested, however, *AMD* processor architecture uses a different subset of kernel flags, which currently are not implemented on *lib viewer*. The platform compatibility of *lib viewer* roadmap should be checked before the IaaS deployment.
- The optimum VM network interface to work with VMDIRAC is an authomatic network configuration with private IP and out-bound connectivity.
- VMDIRAC is able to deal with static network configuration. This can be used for testing purposes, but the maintenance of such approach in production level is problematic, any minor change in the IaaS network configuration, should have the corresponding adjustment in DIRAC Configuration.
- OpenStack floating IP assignment to a previously created VM is not recommendable because there is a gap in the responding time between the time the VM is booted and the time the network interface is available and routable [14].

Moreover, there are different ways to provide the scientific software and tools to be deployed at the VM. This part is related to the contextualization of the next section, but at the same time is a matter of the IaaS provider. For image maintenance reasons and also for performance reasons it is recommended to allocate the required software and tools in a *cvmfs* repository [25] and to setup a site http proxy for the VMs use of the *cvmfs* repository.

III. DIRAC SETUP TO RUN A GENERIC SCIENTIFIC APPLICATION AMONG THE IAAS PROVIDERS

Two main topics are related to DIRAC setup for the VMDIRAC extension: the image and contextualization setup, and the VM horizontal auto-scaling setup, which are the possible policies to create and stop VMs. Once the setup is ready, VMDIRAC is able to create and stop VMs among the IaaS providers and to contextualize those VMs in a transparent manner to provide computing resources to run user jobs. Such jobs can be any scientific software which is able to run decoupled jobs in distributed resources.

A. IMAGE AND CONTEXTUALIZATION SETUP

There are three steps to setup at DIRAC Configuration Server: Running Pods, Images and End-points. VMDIRAC defines the Running Pod as a logical abstraction of a particular running conditions. A Running Pod is matching an Image with the corresponding cloud end-point list to run VMs of such Image. VMDIRAC concept of an Image, is including a *boot image* and, optionally, the contextualization of such image. This approach can deal with *ad-hoc* image ready to run without further contextualization, this image has to be prepared to run in a specific endpoint and a particular DIRAC configuration. Additionally, VMDIRAC can manage context images that use a *golden image* and the necessary information for a particular contextualization method. The use of a *golden image* allows to simplify the image management [26], because all the specifics of the endpoint and scientific application environment is in the contextualization part, while the *golden* image can be distributed to the different IaaS providers without modification. On the latter contextualization, VM is deployed for a particular scientific application environment. HEP has a contextualization approach, namely HEPiX, using CernVM images and contextualization methods supported by OpenStack and OpenNebula. This CernVM approach can also be used with other scientific applications. Instead of a *golden* image depending on the CernVM platform, VMDIRAC also supports a generic *golden* image, which can be configured using a *ssh* contextualization, if an in-bound connectivity is available in the VM for *ssh* and *sftp* operations. A DIRAC image setup can be an *ad-hoc* image or the following contextualized images:

- HEPiX - OpenNebula: DIRAC image context is included in an ISO context image, which has to be previously upload to the IaaS provider to be mounted by the CernVM init process. The end-point context is passed to the VM at submission time. VMDIRAC gets the parameters from the corresponding end-point section and set this environment using the OpenNebula context section, which creates an on-the-fly ISO image, then CernVM mounts it and loads the end-point context.
- HEPiX - OpenStack: DIRAC image context is provided by *amiconfig* tools, sending the scripts in nova 1.1 *userdata*. End-point context is provided through nova 1.1 *metadata*, which is specific for each OpenStack IaaS end-point and selected on submission time from the DIRAC Configuration Server.
- Generic contextualization, using any platform for *golden* image with a *ssh* demon listening in a port with in-bound connectivity in the VM. The VM boots, the VMDIRAC polls the active *sshd* port, runs the DIRAC and the end-point configuration using *sftp* and *ssh* connections.

In this manner, a VM *golden* image is contextualized to deploy DIRAC on multiple IaaS providers, following methods of the industry and research image contextualization, and also considering a generic contextualization valid for any VM images with *ssh* connectivity.

B. VM HORIZONTAL AUTO-SCALING SETUP

VMDIRAC can be configured with different policies for the creation and stoppage of the VMs. Each end-point has associated a VM allocation policy and a VM stoppage policy.

The VM allocation policy can be *elastic* or *static*. The *static* VM allocation is used when a IaaS provider defines a constant number of VM slots that can be accessed. The *elastic* allocation is used to create new VMs when there are jobs queued in DIRAC. For this purposed the Running Pod configuration section has the *CPUPerInstance* option, which defines the minimal overall CPU of the DIRAC jobs waiting in the task queued to submit a new VM. The parameter is used for the tuning of the VM delivery elasticity. Therefore,

 ISBN: 978-1-61208-290-5

a *CPUPerInstance* can be set to a longer time to use the available resources in a more efficient manner, saving creation overheads, and to a shorter time to setup an exhaustive use of the available resources aiming to finish the production in a shorter total wall time, but with higher resource costs due to additional overhead. In regular basis, *CPUPerInstance* references, from shorter to longer values, could be defined to:

- *Zero* to submit a new VM with no minimal CPU in the jobs of the tasks queue.
- A longer value could be the average required CPU of the jobs as a compromise solution between VM efficiency and total wall time.
- A very large value to maximize the efficiency in terms of VM creation overhead, for the cases where the production total wall time is not a constrain.

The VM stoppage policy can be setup to *elastic* or *never*. Elastic policy stops the VM if there are no more jobs running in the last *VM halting margin time*, which is an option to be setup. Anyway, VMs can be stopped by the VMDIRAC admin or by the HEPiX stoppage in the CernVM images (responsibility of each IaaS provider). If a running VM is required to be stopped, then the VM orderly stops, declaring the running job stopped in DIRAC (which can be resubmitted), then halting the VM.

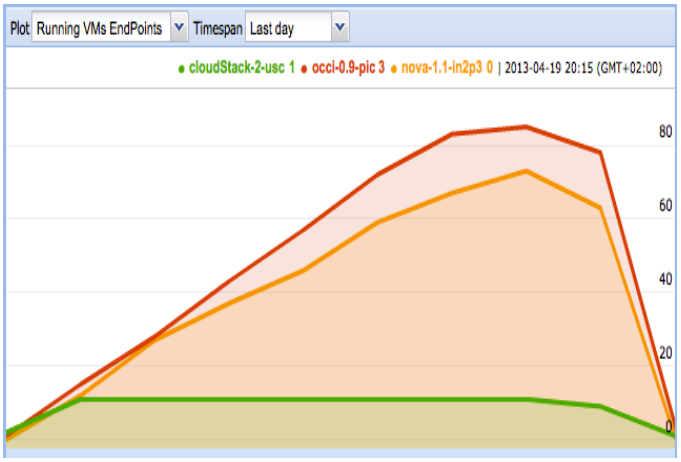


Fig. 2: Running VMs by End-point

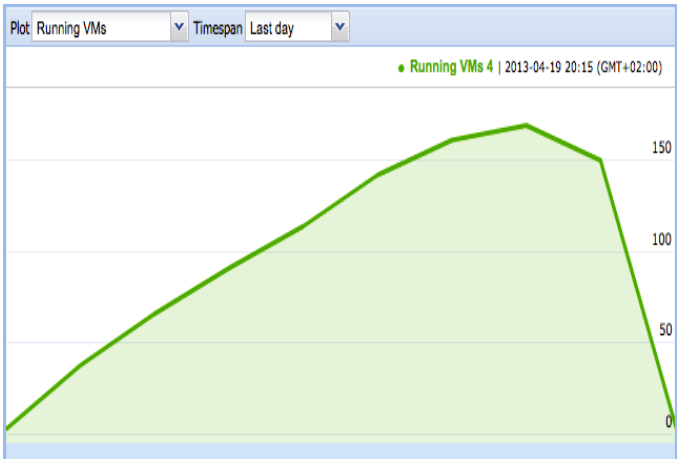


Fig. 3: Overall Running VMs

IV. DIRAC WEB INTERFACE FOR CLOUD MANAGEMENT

DIRAC supports the job management that can be setup to run in Cloud or other distributed resources. This includes a wide number of tools to submit jobs, design workflows for eScience productions, manage execution, plots, accounting and all the necessary for eScience communities [27]. The Configuration Server Web interface allows to setup the Federated Hybrid Cloud as well as different Running Pods, Images and End-points to use transparently such IaaS infrastructure.

VMDIRAC, the DIRAC cloud extension, is also providing a Web interface for the management of the Cloud. There is a VM Browsing to monitor the VMs and take history logs and plots of each VM as it was shown above in Fig. 1. Furthermore, there is a VM Overview to plot the main statistics: running VM by end-point in Fig. 2, overall running VMs in Fig. 3, started jobs, average load, transfer data and transfer files.

Fig. 2 and 3 are plots of the same run. The running application is LHCb simulation of proton-proton collision. The IaaS providers are USC with CloudStack, CC.IN2P3 is supported by OpenStack and PIC by OpenNebula. The VM allocation policy is *elastic* for the three IaaS providers. Contextualization is an *ad-hoc* image with Centos, while PIC and CC.IN2P3 are using CernVM *golden* image and the corresponding contextualization. The VM stoppage policy is *elastic* in the three IaaS providers.

DIRAC configuration has a maximum number of 10 VMs for USC, once the threshold is reached there is a plateau until the end of the workload (green line in Fig. 2). For the case of CC.IN2P3 and PIC IaaS providers the maximum number of VMs has not been reached.

Fig. 3 shows the overall aggregation of the three IaaS providers. VM allocation and stoppage policy is corresponding to a scale-up when there are jobs pending in the DIRAC task queue, and a scale-down when there are no more jobs in the task queue and the VMs workload is finished.

Some of the operations, like the Configuration Server management or the manual VM stoppage, are only authorized

 ISBN: 978-1-61208-290-5

for the VM administrator, who can also monitor all the jobs. A general DIRAC user is authorized to monitor his or her jobs, and the VM Web interface without operate with the VMs.

V. CONCLUSION

This paper has described how to run eScience applications in Federated Hybrid Clouds using DIRAC. This includes instructions and recommendations to the IaaS providers to be aggregated in a federated manner. It also has been defined how to setup the image and context management to run scientific applications, also considering the VM deployment of the scientific software and tools. At the same time, it has been shown the necessity of two roles: administrator of the configuration system and VMs, and the scientific user, who uses DIRAC submitting jobs, which transparently run in Federated Hybrid Clouds.

The deployment of DIRAC portals including the cloud extension VMDIRAC, is a proved tool to aggregate IaaS providers in the level of NGIs supporting multiple VOs, and also in medium and big scientific communities.This schema provides solutions to the deployment and management of SaaS of a wide range of scientific communities, from small communities which can be federated to face the DIRAC portal operations, to big communities who may exploit their own DIRAC portal or integrates in a multiple VO portal. Thus, the proposed strategy is addressing the sustainability through *industrial concentration* of the management of SaaS in Federated Hybrid Clouds, and at the same time allowing *local development* by the aggregation of distributed IaaS resources.

ACKNOWLEDGMENT

PIC is maintained through a collaboration between the Generalitat de Catalunya, CIEMAT, IFAE and the Universistat Autónoma de Barcelona. This work was supported in part by grants of the Ministerio de Educación y Ciencia, Spain: FPA2007-66152-C02-01/02 and FPA2010-21816-C02-01/02, assigned to PIC. Additional support was provided by the EU 7th Framework Programme INFRA-2007-1.2.3: e-Science Grid infrastructures Grant Agreement Number 222667, Enabling Grids for e-Science (EGEE) project and INFRA-2010-1.2.1: Distributed computing infrastructure Contract Number RI-261323 (EGI-InSPIRE).

This work was also supported by projects FPA2007-66437-C02-01/02 and FPA2010-21885-C02-01/02, assigned to UB and USC.

We are greatly in debt with France Federated Cloud, in particular with the *Centre de Calcul* of the IN2P3 for providing part of the VMs used in the presented test results.

References

[1] Lhcb computing with dirac . [Online]. Available: http://lhcb-comp.web.cern.ch/lhcb-comp/DIRAC/

[2] R. Graciani Diaz, A. Casajus Ramo, A. Carmona Aguero, T. Fifield, and M. Sevior, ``Belle-dirac setup for using amazon elastic compute cloud," *Journal of Grid Computing*, vol. 9, pp. 65--79, 2011, 10.1007/s10723-010-9175-7. [Online]. Available: http://dx.doi.org/10.1007/s10723-010-9175-7

[3] France grilles et du cloud . [Online]. Available: http://www.france-grilles.fr/-Presentation-

[4] Bes collaboration . [Online]. Available: http://bes.ihep.ac.cn/

[5] Cta observatory . [Online]. Available: http://www.cta-observatory.org

[6] Linear collider collaboration . [Online]. Available: http://www.linearcollider.org

[7] D. Wentzlaff, C. Gruenwald, III, N. Beckmann, K. Modzelewski, A. Belay, L. Youseff, J. Miller, and A. Agarwal, ``An operating system for multicore and clouds: mechanisms and implementation," in *Proceedings of the 1st ACM symposium on Cloud computing*, ser. SoCC '10. New York, NY, USA: ACM, 2010, pp. 3--14. [Online]. Available: http://doi.acm.org/10.1145/1807128.1807132

[8] A. Gupta, D. Milojicic, and L. V. Kalé, ``Optimizing vm placement for hpc in the cloud," in *Proceedings of the 2012 workshop on Cloud services, federation, and the 8th open cirrus summit*, ser. FederatedClouds '12. New York, NY, USA: ACM, 2012, pp. 1--6. [Online]. Available: http://doi.acm.org/10.1145/2378975.2378977

[9] G. Mateescu, W. Gentzsch, and C. J. Ribbens, ``Hybrid computing - where hpc meets grid and cloud computing." *Future Generation Comp. Syst.*, vol. 27, no. 5, pp. 440--453, 2011. [Online]. Available: http://dblp.uni-trier.de/db/journals/fgcs/fgcs27.html#MateescuGR11

[10] B. Kocoloski, J. Ouyang, and J. Lange, ``A case for dual stack virtualization: consolidating hpc and commodity applications in the cloud," in *Proceedings of the Third ACM Symposium on Cloud Computing*, ser. SoCC '12. New York, NY, USA: ACM, 2012, pp. 23:1--23:7. [Online]. Available: http://doi.acm.org/10.1145/2391229.2391252

[11] France grilles dirac portal . [Online]. Available: http://dirac.france-grilles.fr/DIRAC/

[12] Ibergrid dirac portal . [Online]. Available: http://dirac.ub.es/DIRAC/

[13] Official dirac webpage . [Online]. Available: http://diracgrid.org/

[14] V. Méndez, A. Casajus, V. Fernández, R. Graciani, and G. Merino, ``Rafhyc: An architecture for constructing resilient services on federated hybrid clouds," *Journal of Grid Computing*, 2013.

[15] Nagios infrastructure monitoring . [Online]. Available: http://nagios.org

[16] Ganglia monitoring system . [Online]. Available: http://ganglia.info

[17] Globus gridftp . [Online]. Available: http://www.globus.org/toolkit/docs/latest-stable/gridftp/

[18] Cloud data management interface (cdmi) . [Online]. Available: http://www.snia.org/cdmi

[19] Amazon simple sotorage service (s3) . [Online]. Available: http://aws.amazon.com/es/s3/

[20] A. Alvarez, A. Beche, F. Furano, M. Hellmich, O. Keeble, and R. Rocha, ``Dpm: Future proof storage," in *Computing in High Energy and Nuclear Physics 2012*, 2012. [Online]. Available: http://cdsweb.cern.ch/record/1458022

[21] F. Furano, P. Fuhrmann, R. B. da Rocha, A. Devresse, O. Keeble, and A. A. Ayllon, ``Dynamic federations: storage aggregation using open tools and protocols," in *EGI Technical Forum Book of Abstracts*, 2012. [Online]. Available: https://indico.egi.eu/indico/conferenceDisplay.py/abstractBook?confId=1019

[22] V. Mendez, A. Casajus, R. Graciani, and G. Merino, ``Use case: Running monte carlo lhcb simulations using dirac with egi federated cloud," in *EGI Tehcnical Forum Book of Abstracts*, 2012. [Online]. Available: https://indico.egi.eu/indico/conferenceDisplay.py/abstractBook?confId=1019

[23] V. Méndez, V. Fernández, R. Graciani, A. Casajus, T. Fernández, G. Merino, and J. J. Saborido, ``The integration of cloudstack and occi/opennebula with dirac," *Journal of Physics Conference Serires*, 2013. [Online]. Available: http://iopscience.iop.org/1742-6596/396/3/032075

[24] V. F. Albor, J. J. S. Silva, F. Gómez-Folgar, J. López-Cacheiro, and R. G. Diaz, ``Dirac integration with cloudstack," in *Proceedings of 3rd IEEE International Conference on Cloud Computing Technology and Science (IEEE CloudCom 2011)*, 2011, pp. 537--541.

 ISBN: 978-1-61208-290-5

[25] V. F. Albor, V. Mendez, J. López-Cacheiro, R. G. Diaz, J. J. S. Silva, and T. F. P. na, ``User access to cvmfs software repositories on ibergrid,'' in *IBERGRID, 6th Iberian Grid Infrastructure Conference Proceedings*, 2012, pp. 39--49.

[26] P. Bunic, C. Aguado-Sanches, J. Blomer, and A. Harutynunyan, ``Cernvm: Minimal maintenance approach to the virtualization,'' *Journal of Physics Conference Serires*, 2011. [Online]. Available: http://iopscience.iop.org/1742-6596/331/5/052004

[27] A. Tsaregorodtsev, M. Bargiotti, N. Brook, A. Casajus Ramo, G. Castellani, P. Charpentier, C. Cioffi, J. Closier, R. Graciani Diaz, G. Kuznetsov, Y. Y. Li, R. Nandakumar, S. Paterson, R. Santinelli, A. C. Smith, M. S. Miguelez, and S. G. Jimenez, ``Dirac: a community grid solution,'' *Journal of Physics: Conference Series*, vol. 119, no. 6, p. 062048, 2008. [Online]. Available: http://stacks.iop.org/1742-6596/119/i=6/a=062048

 ISBN: 978-1-61208-290-5

GraphTool - a new System of Graph Generation

Iwona Ryszka, Ewa Grabska
Faculty of Physics, Astronomy and Computer Science
Jagiellonian Universisty
Krakow, Poland
e-mail: iwona.ryszka@uj.edu.pl, ewa.grabska@uj.edu.pl

***Abstract*—The purpose of this paper is to present a new software tool for graph edition and generation. The project focuses on providing a graphical editor for defining different types of graphs and rules describing their transformation. The idea of graph grammar systems has been adopted to extend graph derivation functionality. The system architecture with implementation details is described. To illustrate the features of the tool the examples of a graph building and generation are attached.**

***Keywords*—graph, graph grammar, graph grammar system, tool for graph generation**

I. INTRODUCTION

The concepts of a graph and graph transformations are applied to model structures and also to simulate flows or behaviours. The base definition presents a graph as a set of nodes and a binary relation defined on this set. Each element of the relation being a pair of nodes defines an edge that can be directed or undirected. A graph grammar enables the application of local transformations by means of rules called productions to subgraphs of derived graphs. An order of productions application is determined by so called control diagram. A process of graph generation by means of productions is called a derivation. The possibility of attributes assignment extends the graph model by defining an additional semantic layer.

The basic representation of the graph that includes nodes and edges does not require any dedicated graphical software product. However, taking into consideration the requirements for graph grammars and the process of graph derivation, a new application *GraphTool* [1] has been proposed in order to provide an unified graphical environment supporting graph operations. The motivation for implementing a new tool is to provide the editor for custom types such as composite graphs or layered graphs. Additionally, the new approach for defining a grammar system as a grouping mechanism for graph grammars has been addressed within the project. Furthermore, the *GraphTool* application deals with the area of attributes operations.

Section II provides the overview of existing applications supporting graph defining and automatic rewriting. The purpose of Section III is to present the functionality of the *GraphTool* and it is followed by the Section IV that provides example usage of the application for different types of graphs. The Section V describes the implementation details.

II. RELATED WORKS

There exist a number of tools for generating graphs but they are usually specialized for experts of a particular subject and focus on different areas.

PROgrammed Graph REwrting Systems (PROGRES) [2] offers a visual and operational specification language for defining graphs, transformation rules that is combined with the environment for executing specifications in this language. A transformation rule can be formatted as a simple one when only left-hand and right-hand sides are defined. There is an option to define a combined rule that describe several rules by textual control structures, e.g., loops. Additionally, the framework UPGRADE is available and its purpose is to display the flow of applying graph transformations with some formatting options.

A custom graphical language is also introduced by the *Graph Rewriting and Transformation (GReAT)* and is dedicated for graph transformations in the area of domain-specific modelling languages (DSMLs). The rules specify rewriting operations in the form of a matching pattern (closely related to UML class diagram). The application focuses on model transformations, but there is no verification if the generated graph is correct according to target language.

The area of attributed graphs with the theory on confluence and termination properties is investigated by the *Attributed Graph Grammar System (AGG)* [3]. The application supports attributes of algebraic data types, including Java expressions. The graph derivation process can be both manual and automatic, the intermediate steps are not stored. The GTXL format based on XML is used to export generated graphs.

GRaphs for Object-Oriented VErification (GROOVE) [4] project focuses on the verification of object-oriented systems by means of a graph transformation tool set that uses labelled graphs and single push-out (SPO) transformation rules. There is a component for graphical editing of rules and graphs, a generator responsible for creating temporary graphs during the derivation. Additionally, the Model Checker component is available for verification whether the derived system satisfies specific properties.

III. APPLICATION OVERVIEW

GraphTool is a *What You See Is What You Get (WYSIWYG)* editor. The functional areas that are addressed by the tool cover support for creating different types of graphs, building graph grammars by defining productions and a control diagram. Additionally, the process of graph derivation can be simulated and controlled by the user. As an extension the functionality of building grammar systems is offered.

The base application view is presented in Figure 1. Following working areas can be marked within the view:

1) *GraphTool Navigator* is a component responsible for presenting the structure of grouped objects created by the user. By means of the context menu, new elements can be added.

 ISBN: 978-1-61208-290-5

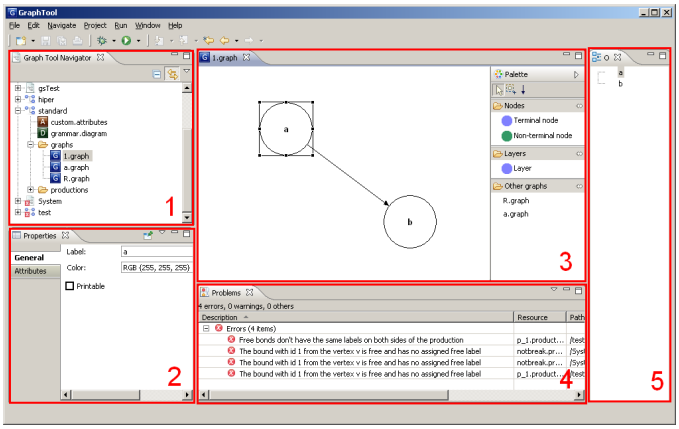


Fig. 1: The main perspective of *GraphTool* application

2) *Properties* is a section providing the functionality of editing properties selected objects in *Graphical editors* area.
3) *Graphical editors* are areas used to build graphs, productions and control diagrams. A dedicated palette with context-specific elements is available.
4) *Problems* represents a supporting section that is used to notify the user about existing issues or warnings automatically detected inside the objects created within graphs.
5) *Outline* is a view showing a hierarchy of objects defined in active *Graphical editiors* area.

A. Graph building process

The compulsory action in the process of creating a graph is to specify a *project*. This entity is used by the application to define a context that will be common for all graphs attached to the given project. Such a specification cover:

- the type of the graph: standard graph, hypergraph (included also the option to create hierarchical graphs [10]), composite graphs [5],
- the type of the edge: directed, undirected.

Building a graph is understood as creating nodes, edges and assigning them labels and optionally attributes. In order to simplify the construction process, a feature to add previously defined graphs within the same project as a subgraph is also available. The folder *graphs* is used to store defined objects.

Graph objects defined with *GraphTool* can be exported to XML files using GraphML format [6]. It supports all types of graphs covered by the tool. Using its flexible extension mechanism to add application-specific data the information about attributes and structure of the graph (sizes and locations of nodes) are also optionally available in the exported file.

B. Attributes

GraphTool application offers the support for attributed graphs. An attribute is defined as a function assigning the value from its domain to an object. Within each project there is a file called *custom.attributes* that is used for creating declaration of attributes that can be later assigned to any object defined within the project. The declaration of a new attribute is understood as specification of an unique name and its domain (the available types are integer numbers, float numbers, strings, enums and arrays). Additionally, the user can specify the default value.

Having defined attributes, their instances can be assigned with values to nodes, edges and graphs. There is a rule that each element of given type and having the same label contains the same attributes.

The application offers an additional functionality in the area of attributes which is called a *template attribute*. During assignment to an element such an attribute it receives a template value (variable). It can be used within a production to represent values which are not known at the moment of creating the production and during the graph derivation they are replaced with with the real one. The next usage of this type of attributes is defining predicates for productions and similarly they are replaced with the current values from a graph when a production is to be applied to this graph. The template attribute can be defined as an expression containing variables including a conditional expression, for example having att_i defined as integer attribute following template attributes are valid:

- $att_i = var_1 + var_2$ interpreted as a sum of values represented by variables var_1 and var_2
- $att_i = (var_1 > var_2)\ ?\ var_2 : var_3$ In case when the value of variable var_1 is greater than var_2, the att_i gets the value of var_2. Otherwise the value of variable var_3 is assigned to att_i.

C. Graph grammars

The concept of the project in the *GraphTool* application is additionally used in the process of defining a *graph grammar* that is specified as a set of productions and a control diagram.

A production entity is represented by two graphs that are called left and right side, respectively. The former describes a subgraph of a derived graph that after application of the production is transformed to the latter. In the *GraphTool* application the user can build a production by defining both sides from the beginning or using graphs available within the project. A folder *productions* is designed to store user's defined productions within *GraphTool* project.

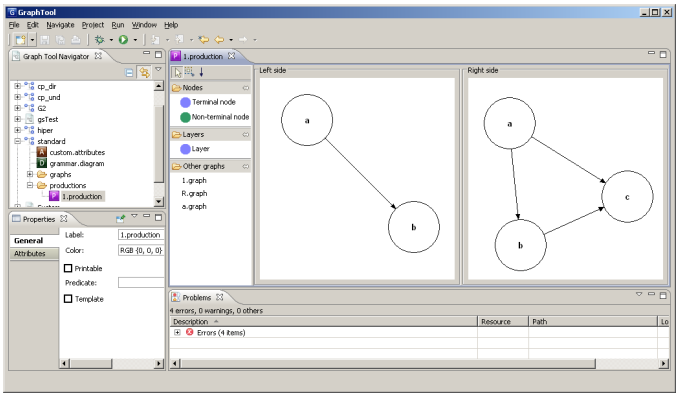


Fig. 2: An example production defined in *GraphTool*

The application offers the feature of an usability predicate for a production. The predicate represents a logical expression and its value is calculated just before application of the production during a graph derivation process. The expression can refer to values of attributes of objects available in the

context of the production: a graph currently derived, its nodes, its edges and also a graph from left side of production.

Having defined transformations, the user builds a control diagram that represents productions' flow. The diagram is a directed graph where nodes represent productions. Additionally, there are two marked nodes called start and stop node that point respectively to the node initializing derivation process and the node completing this process. The diagram is valid when at least one path exists from the start to the stop node. It is stored in the file called *grammar.diagram* within the project.

D. Graph derivation process

In order to derive a new graph, a grammar with a control diagram and an initial graph are needed. Therefore, in the *GraphTool* application the user creates a *configuration* that involves selecting a project and one of the graph available in its *graphs* folder as an initial graph. The process of derivation can be managed by the user by means of selecting next production from the set of available productions according to the control diagram.

The current state of implementation covers the derivation process only for the composite graphs.

E. Grammar systems

As an extension to a graph grammar, the concept of *grammar system* can be introduced. The main purpose of the system is to combine a set of graph grammars and introduce an upper level control diagram that is responsible for switching the context between graph grammars during a graph derivation process. Therefore, the nodes in such a diagram represent graph grammars. Thus, the derivation process is controlled by two diagrams: the grammar system control diagram points to a graph grammar that should be used and derivation follows the control diagram associated with this grammar. When the stop node is achieved, the control is returned to the grammar system control diagram and next grammar can be used. An example derivation within a grammar system is presented in Figure 3.

In the *GraphTool* application the grammar systems can be build from the beginning by defining all graph grammars or by using existing one. The process of graph derivation includes also defining a configuration like in a graph grammar case.

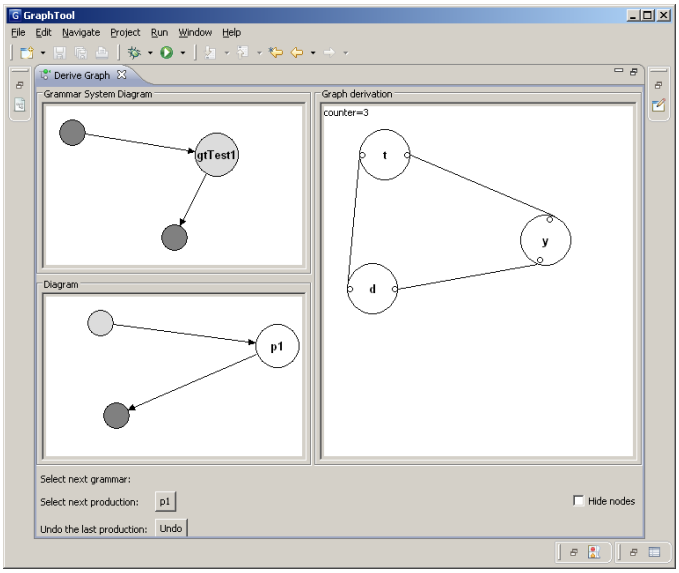


Fig. 3: An example of graph generation process using a grammar system defined in *GraphTool*

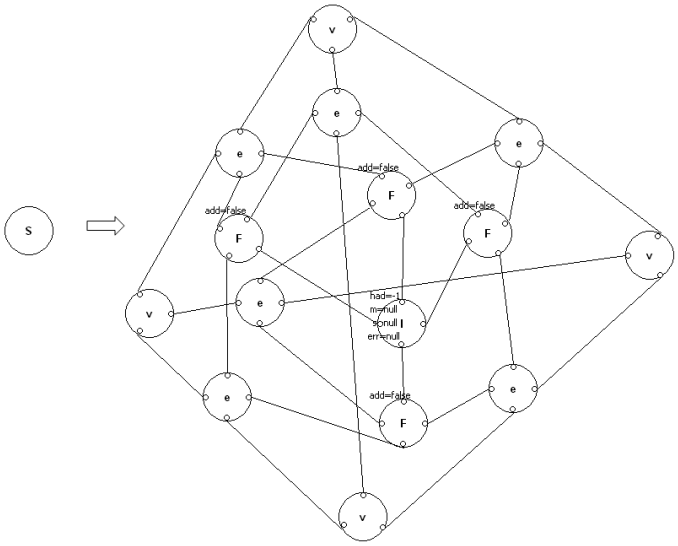


Fig. 4: A mesh generation as a production defined for *h*-modeling finite element method using *GraphTool*

IV. Examples of GraphTool usage

In this section, the examples of advantages of GraphTool for solving computational issues or modeling the structure are presented.

A. Graph grammar system for h-modeling finite element method

There were several attempts to adopt graphs and graph grammars to model *h*-finite element method [5]. GraphTool application has been used as a support tool to construct a grammar system for modeling three dimensional finite element method (3D FEM) with tetrahedral finite elements [7]. Composite, attributed and undirected graphs have been selected to model a finite element mesh.

The system contains a grammar that is responsible for generating the mesh. Its productions represent the mesh transformation. One of them is shown in Figure 4.

The presented production is an example of usage template attributes as the values of integer attribute *fn* can be incremented when the production is applied. By means of the template attributes, the number of productions could be reduced.

The next example of graph grammar within the system contains productions that focus mainly on attributes operations. By means of this process some logic operations such as calculating a solution vector can be performed. Thus, array attributes with variable lengths are useful to store information about such a vector. An example of such a production is shown in Figure 5.

The proposed system proves additionally the advantage of introducing the support for graph attributes. Such objects are used as a *global memory* in the given grammar system. The memory is initialized during defining a graph that is used in

 ISBN: 978-1-61208-290-5

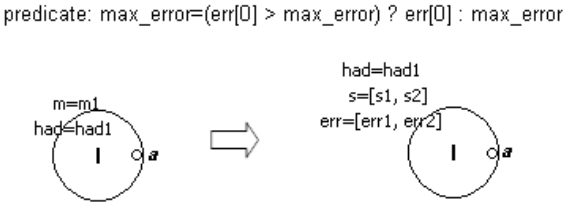


Fig. 5: A production for assigning values to array attributes *s*, *err* for node with label *I* together with a predicate specified using attributes

the derivation process as an initial one. During this process the variables and their current values can be used to calculate values of predicates for productions or values for template attributes. For the grammar system for modeling 3D FEM the memory is defined as two attributes: *error_max* and *accuracy*. For the production shown in Figure 6, the value of its predicate depends on the current value of the attribute *error_max* that is assigned to the generated graph and the value of the attribute of a node in the graph according to the left side of the production.

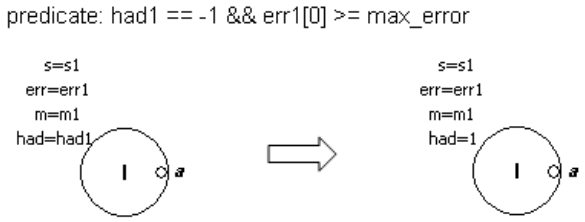


Fig. 6: A production for virtual h-refinement

B. Hierarchical graphs for generating virtual Grids

The Grid can be regarded as the implementation of the distributed computing concept. The structure of the environment that contains grouped components can be modeled by means of hierarchical graphs. The graph derivation process can be used to simulate building the grid environment.

The approach proposed by Lu and Dinda [8] suggests separating topology generation from annotations. The application of hierarchical graphs with attributes for the grid generation was proposed in [9] and the concept has been enhanced with the new structure including layers in [10].

GraphTool offers the environment to construct hierarchical graphs. It can be used to create graphs representing the gird structure together with attributes as presented in Figure 7. The application is able to verify the correctness of hierarchical graphs for example in the area of edges that cannot connect a node with its internal nodes. Additionally, the hierarchy can be built for hypergraphs where both nodes and hyperedges can be nested.

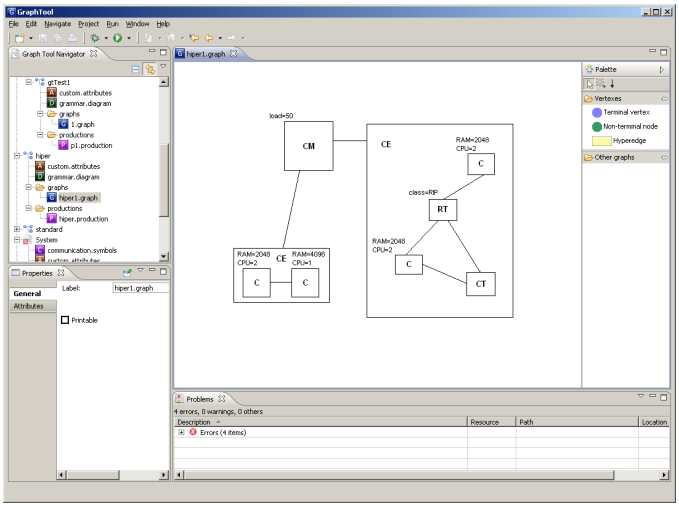


Fig. 7: A hierarchical graph representing a grid

V. Implementation Details

GraphTool is an application based on Java Platform SE 7. It has been created as a plugin for Eclipse Integrated development environment. The required release is at least Eclipse Helios version 3.6.1, available at [11]. The tool can be also shipped as a standalone rich client application. Supported operating systems are both Windows and Unix based and the only additional requirement is an installed Java Runtime Environment in 1.7 version, available at [12].

The plugin uses several open sources libraries. The first is Graphical Editing Framework technology [13] that is regarded as a middleware for providing graphical layer within Eclipse editors and views. For transforming objects between Java and XML JAXB version 2.2.3 is used. Additionally, JUNG library [14] in version 2.0.1 is used to model basic structure of graphs and perform operations on them such as finding shortest path.

The current state of implementation covers the functionality for deriving graphs using both graph grammars and graph grammar systems for the composite graphs. This type of graph uses a constant embedding transformation during a production appliance. Therefore, there is no need to specify any additional conditions for the production. The character of the composite graphs add additional requirements for finding a subgraph during applying a production when the bonds of the edge have to be checked. Additionally, the parsing and calculating values of the expression in attributes JavaScript engine is used.

VI. Conclusion

In this paper, a new software application for editing and generating graphs has been presented. The design of the tool with the focus on task-oriented modules results in the support for creating different type of graphs, including attributed ones and productions for their generation. The concept of template attributes has been also described. By means of grammar systems the process of creating new graphs has been enhanced. The functionality of the GraphTool application has been illustrated on the example concerning defining graph grammar systems for modeling three dimensional finite element method and the hierarchical graph representing the grid environment.

Further research in the application development can cover

 ISBN: 978-1-61208-290-5

support for grammar systems where particular grammars represent different types of graphs. Such a feature introduces a possibility of exploration in the area graphs transformations between specific representations. The idea of templates for attributes can be a source for further investigation as it enables to add logic to the graph structure and during the graph derivation process some calculations can be performed.

REFERENCES

[1] I. Ryszka, "Implementation of graphs model generation and their appliance", PhD Thesis, Jagiellonian University, in press.

[2] A. Schürr, A. J. Winter and A. Zündorf, "Graph Grammar Engineering with PROGRES", ESEC 1995, pp. 219-234.

[3] G. Taentzer, C. Ermel, and M. Rudolf, "The AGG approach: Language and tool environment" in "Handbook of Graph Grammars and Computing by Graph Transformation", volume Volume 2: "Applications, Languages and Tools. World Scientific", 1999, pp. 163–246.

[4] A. Rensink, "The GROOVE Simulator: A Tool for State Space Generation. In: Applications of Graph Transformations with Industrial Relevance (AGTIVE)", Lecture Notes in Computer Science 3062, Springer 2004, pp. 479-485.

[5] A. Paszyńska, E. Grabska and M. Paszyński, "A Graph Grammar Model of the hp-adaptive Three Dimensional Finite Element Method. Part I", Fundamenta Informaticae, 114(2), 2012, pp. 149 – 182.

[6] GraphML, http://graphml.graphdrawing.org, September 2013

[7] I. Ryszka, A. Paszyńska, E. Grabska, M. Paszyński and M. Sieniek, "Graph grammar systems for modeling three dimensional finite element method", in press

[8] D. Lu and P. A. Dinda, "GridG: Generating Realistic Computational Grids", SIGMETRICS Performance Evaluation Review, Volume 30, num 4, 2003, pp. 33-40.

[9] B. Strug, I. Ryszka, E. Grabska and G. Ślusarczyk, "Virtual 'Computational Grid by Graph Transformations", F. Davoli et al. (eds.), Remote Instrumentation for eScience and Re-lated Aspects, Springer Science + Business Media, LLC 2012.

[10] E. Grabska, W. Palacz, B. Strug and G. Ślusarczyk, "A Graph-Based Generation of Virtual Grids, 9th International Conference on Parallel Processing and Applied Mathematics (PPAM 2011)", Lecture Notes in Computer Science 7203, 2012, pp. 451 – 460.

[11] The Eclipse Foundation open source community, http://www.eclipse.org, September 2013

[12] Java Technology, http://www.java.com, September, 2013

[13] GEF - Graphical Editing Framework for Eclipse, http://www.eclipse.org/gef, September 2013

[14] JUNG - Java Universal Network/Graph Framework, http://jung.sourceforge.net/, September 2013

 ISBN: 978-1-61208-290-5

Supporting Coding Activity by Associating Web Bookmarks With Source Code Features

Ken Nakayama
Institute for Mathematics and Computer Science
Tsuda College
2-1-1 Tsuda-machi, Kodaira-shi
Tokyo, Japan
e-mail: ken@tsuda.ac.jp

Eko Sakai
Department of Humane Informatics
Faculty of Letters
Otani University
Koyama-Kamifusacho, Kita-ku
Kyoto, Japan
e-mail: echo@res.otani.ac.jp

Yoshihisa Nitta
Department of Computer Science
Faculty of Liberal Arts
Tsuda College
2-1-1 Tsuda-machi, Kodaira-shi
Tokyo, Japan
e-mail: nitta@tsuda.ac.jp

***Abstract*— Computing tools are often provided as various kinds of software libraries. To enjoy the benefit of the latest technologies, a user has to continuously learn their usage. During such learning activity, referring to various web articles is a common practice for programmers. Although a variety of information can be found on the web, specific information of interest tends to exist fragmented and scattered on many web pages. Therefore, bookmarking them in a well-organized way is inevitable for later use. However, manually organizing bookmarks requires extra effort for a user. To overcome the problem, a semi-automatic bookmark manager for coding related web pages is presented. A prototype system is implemented as a plug-in of an integrated development environment. The system observes bookmarking activity by a user on a web browser, and associates each bookmark with (1) features of the source code being edited, and (2) features of the source code editing in current session. The target language of prototype system is Java. User experience of the prototype is presented as preliminary evaluation.**

Keywords—programming; web bookmark; source code feature

I. INTRODUCTION

For programmers, user-contributed contents on the web, such as blogs, are major source of information about coding. It is common to use an Integrated Development Environment (IDE) and a web browser together. The coding and web referencing activities are indivisible. There exists various useful knowledge which is not included in reference manuals. For example, early adopters of some software library may report bugs on the latest version, or senior programmers may post programming tips for beginners, and so on. The fact that a large number of users continue to contribute a wide variety of articles at all times, is the source of the strength of those articles. As a result, user-contributed contents cover wide range of topics as a whole.

The fact, however, is also the cause of drawbacks. Since such articles tend to be short, unorganized, and possibly incorrect, almost all of them are not suitable for a programmer's specific situation and purpose. Therefore, the programmer has to search and gather useful web pages with effort from his own viewpoint.

As web pages that have been judged valuable are likely to be viewed again by the programmer in the future, they should be bookmarked to reduce efforts of searching them again. But in practice, coding-related bookmarking is not so easy for a programmer. Commonly used conventional tree-structured bookmarking seems to be too simple to express semantically mutually related pages. Moreover, a programmer may want to classify web pages from a viewpoint of a specific problem solved. This makes the semantic structure more complex. In other words, the semantic relation and the viewpoint of the problem solved are not necessarily orthogonal.

Although more sophisticated bookmarking methods such as the use of tags, for example [1], (this is equivalent to putting one bookmark into two or more folders) may be able to express the structure, they often need a cumbersome operation to use in practical programming. There is an attempt [2] to integrate Stack Overflow (via [3]) crowd knowledge in the IDE, but it is desirable that arbitrary web pages can be registered to bookmark.

In this paper, semi-automatic tag-based bookmarking system for coding activity is presented. Observing both an IDE and a web browser, the system extracts features (tags) from the source code being edited when a new bookmark is registered. There are some literatures, for example [4], that attempt to analyze programmer's action on IDE, but still little is known. Thus, the prototype system presented here adopted a simple static method. The bookmark is added to the bookmark table in the system together with features. Later, a programmer can retrieve recommended bookmarks from the tagged bookmarks using another source code as a query.

A prototype system has been implemented as a plug-in of IDE eclipse (via [5]). The target language is Java. The primary user interface consists of three extra buttons (`start`, `end`, and `search`) added to the IDE. The feature used in current prototype system is identifier (e.g. class name, method name) which appears in the source code.

In the next section, to introduce our motivation, coding-related bookmarks "badly" organized by a nonprofessional programmer are reviewed. We believe that most of ordinary programmers have the similar problem in handling bookmarks. Section III describes a prototype system [6] from a programmer's point of view. In Section IV, the model of bookmark registration with source code feature and edit feature is presented, then the mechanism of bookmark retrieval is explained in Section V. There are some implementation issues

 ISBN: 978-1-61208-290-5

in Section VI. Since the evaluation of the system is still under way, preliminary user experience and discussion is presented in Section VII. Finally, Section VIII concludes this paper.

II. CODING-RELATED BOOKMARKS

A. Original Bookmark List

We interviewed one nonprofessional programmer and have analyzed his bookmarks. The purpose is to grasp how the bookmark of the Web page seen during coding is classified. There were 870 programming-related unique bookmarks in the bookmarks added during the period of about two years and three months .

These were only saved in order of the addition. That is, there was no subfolder. The programmer explained the reason as follows.

- Most of these bookmarks were added during coding. During coding, he had to be concentrated on the coding activities itself. And he felt it troublesome to classify bookmarks after coding. Anyway, the bookmark was not classified.
- If bookmarks were arranged in the added order, he thought that the relevance between bookmarks could be guessed by the nearness of a time stamp. However, when the bookmark list became larger than a screen, manual search became difficult gradually. Since only the title of the Web page was saved in the bookmark list, word search was useless.

After all, he did not use the bookmark list effectively.

B. Session Segmetation

It can be considered that the bookmarks with near addition time are for the same or similar purpose. In this analysis, a series of bookmarks added within 3 hours after the last registration were defined as one session. As a result, 870 bookmarks were divided into 332 sessions. That is, it is assumed that these bookmarks were added for 332 individual purposes. During one session, 2.6 bookmarks were added on the average. The maximum was 24 bookmarks per session.

C. Free Tagging Experiment

Next, he was asked to give tags freely in order to know how the programmer recognizes each bookmark. He was instructed to give tags completely in order to make it easy to look for a bookmark for himself in the future. He was allowed to give two or more tags like ”Java / IDE / Eclipse/JDT” to the single bookmark. As a result, 63 kinds of unique tags were used. It means that 2.8 tags are contained on the average in one session (this corresponds to one purpose). The maximum was nine tags per session. Many of relations between tags were a main classification / ”sub classification” types, such as ”programming language/Scala”, for example. However, there were some relations between the tags which lack consistency. For example, although ”Java” is programming language, ”programming language” tag is omitted.

The following comments were obtained from the programmer through the interview.

- ”Java” appeared repeatedly and it was obvious that it was ”programming language.”
- Based on the estimated number of the search results by tags, he decided the criteria ”how detailed tags should be given.” The number of the bookmarks which he can look through easily is 20-30. He thought that the number of search results should not exceed this.
- He wanted to search bookmarks, using an error message, a method name, or a class name as a query. However, it was difficult to realize this manually.

III. PROTOTYPE SYSTEM

A. User Interface

From a programmer’s point of view, the system has two phases: **coding and bookmark addition** and **bookmark reference**.

The primary user interface is three extra buttons (`start`, `end`, and `search`) added to an IDE as shown in the left window of Fig. 1 (buttons are shown together with a lot of other buttons). `Start` and `end` buttons are used in the former phase to let the system know when a bookmark session starts and ends. On the other hand, `search` button is used to view recommended bookmarks in the latter phase.

B. Coding and Bookmark Registration

This phase is the same with ordinary coding and bookmarking activity except that the programmer explicitly indicates the semantic sections of editing and bookmarking activities using ”`start`” and ”`end`” buttons to the system. The system assumes a source code is being opened on IDE. A programmer is supposed to click `start` button before he starts editing with some specific intention. Once it is done, he clicks `end` button. Activities between `start` and `end` is a bookmark session. The granularity of a bookmark session is up to the programmer.

When a programmer finds some Web pages useful, he can add bookmarks to the web browser as usual. When the bookmark session ends, all bookmarks added during the session are associated with features of the source code and editing activity during the bookmark session.

Current implementation of the system requires clicking `end` button on completion of one bookmark session even if there is no bookmark registered during the session. By clicking `start` button again, the programmer may start the next bookmark session.

C. Bookmark Retrieval

When a programmer clicks the `search` button with a source code opened on the IDE, the system recommends bookmarks as shown in Fig. 2, if any, based on the code. Recommended bookmarks are listed together with “Changed Methods” and “Changed Classes” that represent edit features. The model for each phase is detailed in the following.

 ISBN: 978-1-61208-290-5

Fig. 1: `Start`, `end`, and `search` buttons on an IDE.

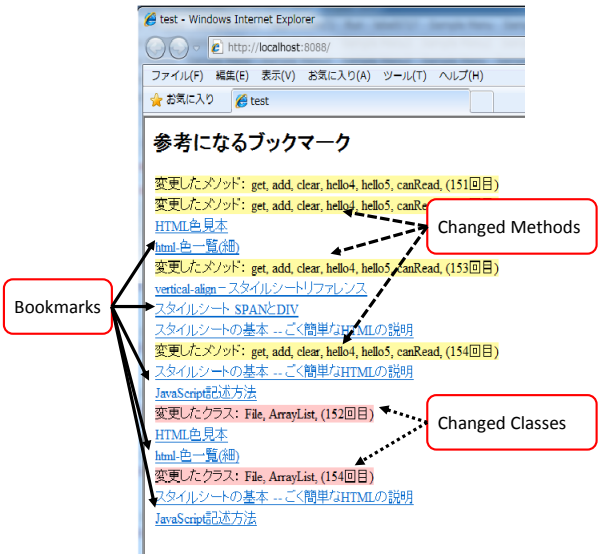


Fig. 2: Recommended bookmarks (shown on the Web browser).

IV. Bookmark Registration Using Features of Code and Coding Activity

A. Bookmark Session

A programmer usually breaks down their task into semantically coherent coding activities. “Coherent” in this context means that each activity reflects programmer’s specific editing intention. If a bookmark can be associated with a coding activity, and if features of such activity are obtained, these features can be used for bookmark classification.

To split a series of programmer’s coding interactions into semantically coherent coding activities, we ask a programmer explicitly indicate the beginning and the end of each activity. We call the period between them **bookmark session** (regardless of whether any bookmark is registered or not during the period). Bookmarks that are registered between them are associate with the corresponding coding activity.

Fig. 3 illustrates the bookmark registration in coding and bookmark addition phase. The horizontal right arrow depicts the operation time flow. Interaction and bookmark of a Web browser (Google Chrome) are shown above the arrow, while interaction and stored features of an IDE are shown below. The time period explicitly indicated by a programmer using `start` and `end` buttons is a bookmark session.

B. Source Code Feature and Edit Feature

To make such classification useful, the followings are essential: (a) the definition of features, (b) the method to infer features from a coding activity, (c) the definition of query types, and (d) the method to get matching bookmarks with a query.

Since the source code being edited may erroneous, even syntactically incomplete, features should be obtained without requiring running it. Query should be as simple as possible for a programmer in order not to disturb coding activity.

We first define **source code feature** which reflects the source code being edited or browsed. The source code feature is not directly used when registering a bookmark. Rather, we define **edit feature** as the difference of source code features at

 ISBN: 978-1-61208-290-5

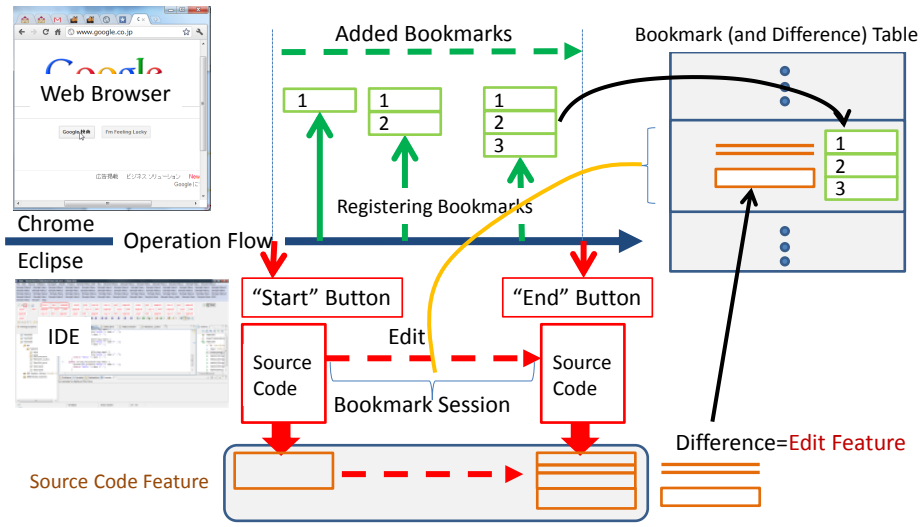


Fig. 3: The overview of the propsed system (coding and bookmark addition).

the beginning and the end of a bookmark session. We expect the edit feature reflects a programmer's coding activity.

Inherently, the best way should be determined experimentally. Various methods exist for this purpose. For example, the system may be track the cursor position or the editing point which a programmer operates. However, such real-time tracking may be difficult to realize efficiently, and may also have a problem of execution performance. Therefore, as a starting point, we have chosen simplest way for the prototype system.

C. Features Used in The Prototype

Examples of source code feature and edit feature are shown in Fig. 4.

Source code feature $F_S(c)$:
A set of the following names (identifiers of Java programming language) of a Java source file c . (1) class names and method names defined, (2) instanciated class names, and (3) method names used. The occurrence frequency (the number of appearance in the code) of each name is not used. The system statically (syntactically) analyze the code to gather these names.

Edit feature $F_E(c_1, c_2)$:
Set difference of the source code features before and after a bookmark session, that is the sum-set of both "the added name" and "the deleted name"

$$F_E(c_1, c_2) = (F_S(c_1) \cup F_S(c_2)) - (F_S(c_1) \cap F_S(c_2)) \quad (1)$$

where c_1 and c_2 are the content of the source file at the begininng and the end of a bookmark session, respectively.

D. Registering Bookmarks With Features

At the end of a bookmark session, bookmarks registered during the session are associated with edit feature, and this is added to the bookmark table (Fig. 3).

E. Behavior of The Prototype

When the `start` button is clicked, with the source code content c_1 being opened, the system performs the followings:

1) The current source code feature $F_S(c_1)$ is recorded.
2) In order to detect the bookmarks added during the editing, the bookmark list of the time is recorded. That is, already registered bookmarks are not taken into account. Such bookmarks are not shown in Fig. 3 for simplicity.

After that, a programmer may edit the source code. When a programmer thinks that he completed one semantic section of an editing, he clicks the `end` button. Suppose that the content of the source code is now c_2. Activities from `start` through `end` is a bookmark session. The system performs the followings:

1) The current source code feature $F_S(c_2)$ is recorded.
2) The edit feature $F_E(c_1, c_2)$ is calculated using recorded $F_S(c_1)$ and $F_S(c_2)$.
3) Let B be a set of bookmarks that are registered during this bookmark session. This set may be empty. The pair
$$S = (F_E(c_1, c_2),\ B) \quad (2)$$
which represents bookmarks tagged with features, where S corresponds to a bookmark session, is added

 ISBN: 978-1-61208-290-5

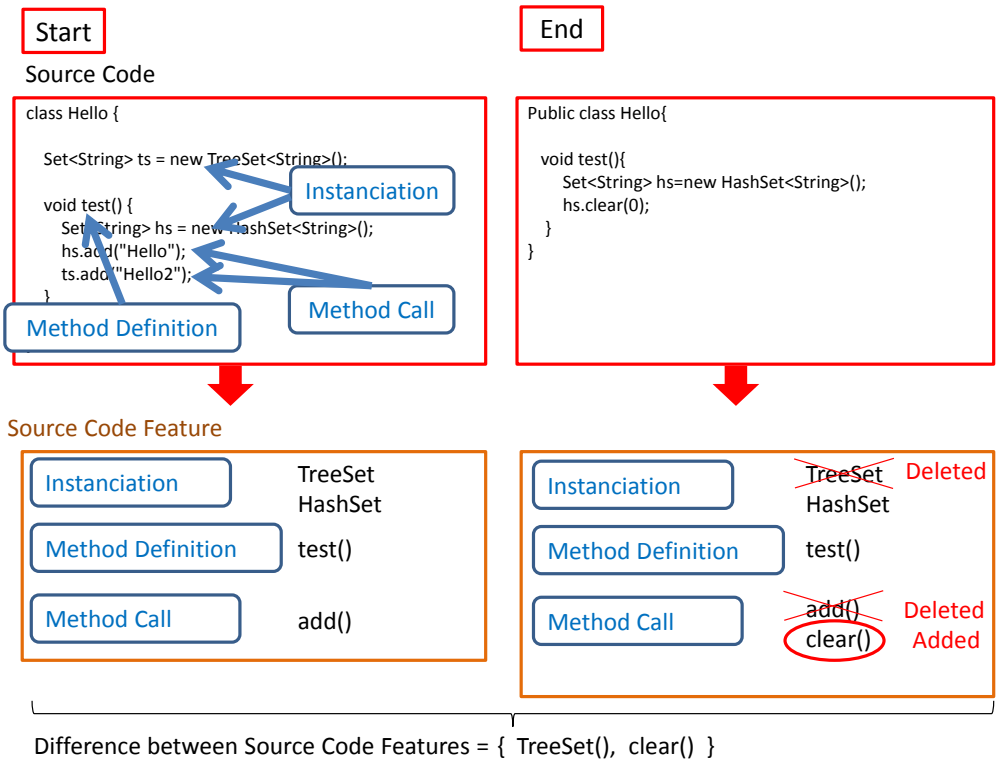


Fig. 4: Souce code feature and edit feature for the prototype.

to the bookmark table. Currently, the source code features at the time of start and end are not recorded.

V. Bookmark Retrieval Using Features

When `search` button is clicked, the recommended bookmarks are retrieved using a source code feature, not a edit feature, as a query. This is a bookmark reference phase. Suppose that N bookmark sessions

$$S_r = (F_{Er}, B_r)\ (r \in 1, \cdots, N) \quad (3$$

are stored in the bookmark table, and the current content of the source code in IDE is c. The source code feature $F_S(c)$ is compared to all stored edit features $F_{Er}\ (r \in 1, \cdots, N)$ to find the maximum match $S_k = (F_{Ek}, B_k)\ (k \in 1, \cdots, N)$ then it is presented to a programmer through a web browser (Fig. 2). This is illustrated in Fig. 5.

In the current prototype system, the similarity metric between features `match` is defined as the number of common elements as follows:

$$\text{match}(F_1, F_2) = |F_1 \cap F_2| \quad (4)$$

where F_1 and F_2 are both features.

VI. Implementation

This prototype system is implemented using Java as a plug-in of Eclipse IDE (via [5]). SQLite relational database (via [7])

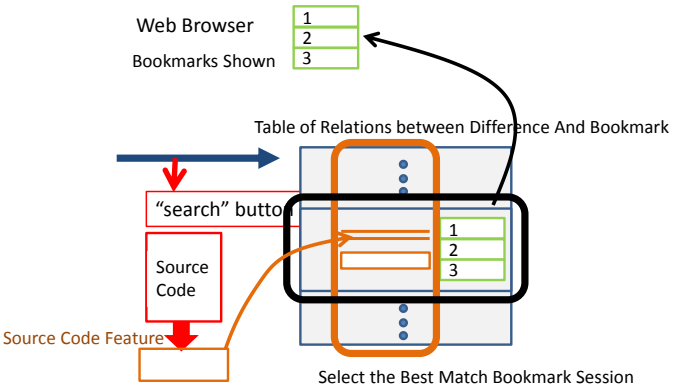


Fig. 5: The overview of the propsed system (bookmark search phase).

is used for the bookmark table. The system reads the bookmark file of Chrome (via [8]) (web browser) directly from a file system. The bookmark file is represented in JSON format (via [9]). For presenting the recommended bookmarks to a programmer, the system has a simple HTTP server as a part of the IDE plug-in.

 ISBN: 978-1-61208-290-5

The source code feature is extracted from the abstract syntax tree (AST) which represents the syntactic structure of a source code. The node of AST recursively represents syntax elements, such as method declaration or a method call.

The system uses AST generated by Java Development Tools (via [10]) (JDT) in eclipse. In JDT, each AST node is an instance of a `ASTNode` class. By defining the subclass of the `ASTVisitor` class, all the `ASTNode` can be scanned in order (or "visited"). In the subclass, `visit` methods with various AST node parameter types can be defined. A desired process can be described to the AST node class by which a `visit` method is defined.

VII. Discussion

A. Edit Feature and Source Code Feature

In the prototype system, a source code feature is used as a query for bookmark reference, while an edit feature is recorded on a bookmark addition. In this sense, the system is asymmetric.

This reflects our naive intuition in the early stage of our design. The edit feature is more specific than the source code feature. At the time of the addition of a bookmark, the more specific feature is desirable. On the other hand, to refer to the bookmark, the more robust feature is desirable. By using source code feature as a query, a programmer can query bookmarks even before editing anything, for instance, just after opening a souce file. This of course can be extended to use edit feature as a query.

Our experience shows that the definition of current edit feature seems to be too specific, because only deleted or newly created methods or classes are recognized as an edit feature. That is, modifications within the definition of existing method or class are not taken into account. The better source code feature and edit feature should be explored.

B. When Should Start and End Be Clicked?

Ideally, the bookmark session should reflect a programmer's subjective semantic chunk of an editing activity. Relying on explicit clicks of `start` and `end` seems to be working well to some degree. However, always forcing a programmer to click these buttons is an extra burden. The means to infer a bookmark session should be explored. At the same time, some way to explicitly control the recognition of a bookmark session should be provided.

C. Problems of Current Implementation

- The class name and method name used as the source code feature are not always a fully-qualified name (FQN) like `java.util.String`. If JDT provides FQN, the system uses it. If FQN is not available, a simple name (without qualification) is used. This may introduce name confusion if the same name is used in more than one context. For example, if the methods of the same name are defined in two or more classes, the system cannot distinguish them to each other.
- When extracting a feature, the order or the frequency of occurrences of a class name or a method name are not taken into consideration. This could be improved.
- In the current system, both the added method names and the deleted method names are included in the edit feature. Distinguishing these sets of names might be informative.

VIII. Conclusion and Future Work

This paper has presented a semi-automatic tag-based bookmarking system for coding activity. A prototype system has been implemented as a plug-in of Eclipse IDE. The target language is Java. Used with a web browser, the system offers a way to register and retrieve personal bookmarks of a programmer. Observing both an IDE and a web browser, the system extracts features (tags) from the source code being edited when a new bookmark is registered. The bookmark is added to the bookmark table in the system together with features. Later, a programmer can retrieve recommended bookmarks from the tagged bookmarks using another source code as a query.

The problem of inferring programmer's intention by observing coding activity is quite difficult. The current prototype system can infer the intention very roughly using simple definition of source code feature and edit feature. However, experience with the system suggested that even such rough intention might be useful for bookmarking. The better features should be explored.

Another direction of improvement is multiuser collaboration. The prototype system assumes the use of a single programmer, but the bookmark table may be shared and collaboratively used by multiple users. For example, collaborative code reading is a promising situation. Suitable method and effectiveness evaluation should be further explored.

References

[1] C. Marlow, M. Naaman, D. Boyd, and M. Davis, "HT06, tagging paper, taxonomy, Flickr, academic article, to read," in *Proceedings of the seventeenth conference on Hypertext and hypermedia*. ACM, 2006, pp. 31–40.

[2] L. Ponzanelli, "Exploiting crowd knowledge in the IDE," Ph.D. dissertation, Master's thesis, University of Lugano, 2012.

[3] Stack Overflow, "Stack Overflow," [retrieved: Jul 28, 2013]. [Online]. Available: http://stackoverflow.com/

[4] A. J. Ko, B. A. Myers, M. J. Coblenz, and H. H. Aung, "An exploratory study of how developers seek, relate, and collect relevant information during software maintenance tasks," *IEEE Transactions on Software Engineering*, vol. 32, no. 12, p. 971, 2006.

[5] The Eclipse Foundation, "Eclipse integrated development environment," [retrieved: Jul 04, 2013]. [Online]. Available: http://www.eclipse.org/

[6] C. Tanaka, K. Nakayama, Y. Nitta, and E. Sakai, "Programming assistance by associating web bookmarks with source code difference," vol. 111, no. 470, Mar. 8–9, 2012, Technical Report of IEICE LOIS2011-111, pp. 231–235, (in Japanese).

[7] SQLite Consortium, "SQLite," [retrieved: Jul 04, 2013]. [Online]. Available: http://www.sqlite.org/

[8] Google, "Chrome," [retrieved: Jul 04, 2013]. [Online]. Available: http://www.google.com/chrome/

[9] JSON.org, "Introducing JSON," [retrieved: Jul 04, 2013]. [Online]. Available: http://www.json.org/

[10] The Eclipse Foundation, "Eclipse Java development tools (JDT)," [retrieved: Jul 04, 2013]. [Online]. Available: http://eclipse.org/jdt/

 ISBN: 978-1-61208-290-5