Big Data Paradigm Developed in Volunteer Grid System with Genetic Programming Scheduler

Jerzy Balicki, Waldemar Korłub, Julian Szymanski, and Marcin Zakidalski

Faculty of Telecommunications, Electronics and Informatics,
Gdansk University of Technology, Gdask, Poland
{balicki,julian.szymanski}@eti.pg.gda.pl, waldemar.korlub@pg.gda.pl,
mzakidalski@gmail.com

Abstract. Artificial intelligence techniques are capable to handle a large amount of information collected over the web. In this paper, big data paradigm has been studied in volunteer and grid system called Comcute that is optimized by a genetic programming scheduler. This scheduler can optimize load balancing and resource cost. Genetic programming optimizer has been applied for finding the Pareto solu-tions. Finally, some results from numerical experiments have been shown.

Keywords: big data, volunteer computing, genetic programming.

1 Introduction

It is estimated that 2.5 exabytes of digital data are captured per day. A collection of large data sets requires some advanced database management tools based on artificial intelligence techniques to allow decision making, discovery and process optimization. Especially, big data sharing is a scientific and practical challenge due to some rapid progresses in finance, business as well as web banking. It is worth to mention that large data sets are gathered by ubiquitous smartphones, tablets, and wireless sensor networks with cameras or microphones. In result, the data store capacity has approximately doubled every three years since the 1980s. Moreover, data storage and also their visualization, analysis and search are still considered as an open problem to solve, too [17].

Massively parallel software on thousands of servers is required and that is why big data (an acronym BD) is not convenient to most relational database management systems. In such systems as desktop statistics and visualization packages, sizes of data are beyond the capability of commonly used tools within a tolerable elapsed time. A single big data set consists of terabytes of data and it can increase to achieve many petabytes for one volume. What is more, progress in speed of data in and out gives an opportunity to take advantage for big data development. Another criterion is wide variety data that is related to a huge range of data types and sources. Above four criteria: high volume, extraordinary velocity, great data variety, and veracity create the 4Vs model for big data description [20].

L. Rutkowski et al. (Eds.): ICAISC 2014, Part I, LNAI 8467, pp. 771–782, 2014.

We can distinguish some differences between big data and business intelligence as regards data use. Some nonlinear system identification methods and inductive statistics are applied for BD to deduce causal effects, nonlinear relationships. We can use regressions to discover dependencies and to find behaviors and predictions. On the other hand, some descriptive statistics can be developed for business intelligence to identify quantity effects or trends.

Genetic programming starts from a goal to be achieved and then it creates an application autonomously without explicitly programming [14]. To some extent, it replies the question that has been formulated by Arthur Samuel - a founder of machine learning - "How can computers be made to do what needs to be done, without being told exactly how to do it?" [18]. This paradigm uses the principle of selection, crossover and mutation to obtain a population of programs. It has been successfully applied to some problems from different fields [15]. Especially, multi-criterion genetic programming (MGP) can determine the Pareto-optimal solutions [2].

In this paper, MGP has been applied as a multi-objective scheduler for efficient using big data by volunteer grids. This scheduler optimizes both a workload of a bottleneck computer and the cost of the system. Moreover, an immunological system based procedure has been applied to handle admissible solutions. Finally, some outcomes for numerical experiments have been presented.

2 Multiagent Approach to Big Data Acquisition and Mining

Big data introduces a lot of issue in terms of data acquisition, storing and mining. Data is often gathered from multiple sources, which may be heterogeneous and spread geographically across the world. Moreover, the collected data may be stored in multiple geographically spread facilities as well due to sheer requirement of storage capacity, which cannot be fulfilled by a single outpost. Like in every distributed system, possibilities of communication loss and node downtime are undeniable and such occurrences need to be handled by software involved in data mining. This problem is even more important in case of mobile settings (e.g. mo-bile sources of data) as availability of data depends on time in such environments. Because of that, connection losses are no longer an anomaly they become a given trait of the system [5].

Multiagent systems are well suited for big data acquisition because of traits, which are commonly assigned to agents. The most important is mobility, which means the ability to move between different facilities. By doing that agents can get closer to the source of data or closer to the data they are about to process. It reduces bandwidth requirements and delays caused by network communication over long distances [5].

The ability to react upon sudden changes of the environment and to act proactively are other important traits, which an agent can take advantage of to improve data mining efficiency. Those traits provide foundation for handling changes in availability of data sources or collected data. Proactivity and autonomy translate to capability of an agent to set its own goals and act upon them

without external influence or control. An agent can proactively decide to move to another set of data or initiate communication with other agents when it sees it feasible. It is especially important in case of big data mining, as the expected results of the extraction of deeply concealed knowledge from the data set, which is processed, cannot be pre-determined. This means that appropriate actions and intermediate goals of the knowledge extraction cannot be predetermined as well, so the agent needs to decide what to do on its own.

Other useful traits of agents include abilities to communicate and negotiate. In agent-based data mining system it is possible to distinguish different roles and groups of tasks that constitute the whole mining process [12]. Individual roles can be then assigned to agents. Through communication and negotiation working groups of agents can be established, each of them containing agents with a unified incentive to fulfill goals of their group.

Agent-based approach can improve efficiency of data mining compared to centralized approaches [23]. It was applied in different domains showing promising results for further research, e.g banking and finance domain [16] or resource allocation in distributed environments [5].

3 Genetic Programming and Immunological Systems

Genetic programming permits discovering a game playing strategy and can be applied in optimal control, planning and sequence induction [14]. Fig. 1 shows an example of a tree as a model of the computer program performance. This tree is equivalent to the parse tree that most compilers (parsers) construct internally from a computer program source. A parse tree consists of branches and nodes: a root node, a branch node, and a leaf node. A parent node is one which has at least one other node linked by a branch under it. A child node is one which has at least one node directly above it to which it is linked by a branch of the tree.

The size of the parse tree is limited by the number of nodes or by the number of the tree levels. Nodes in the parse tree are divided on functional nodes and terminal ones. A functional node represents the procedure randomly chosen from the primary defined set of functions:

$$\boldsymbol{\mathcal{F}} = \{f_1, \ldots, f_n, \ldots, f_N\} \quad . \tag{1}$$

Each function should be able to accept, as its arguments, any value and data type that may possible be returned by the other procedure [14]. Moreover, each procedure should be able to accept any value and data type that may possible be assumed by any terminal in the terminal set:

$$\boldsymbol{\mathcal{T}} = \{a_1, \ldots, a_m, \ldots, a_M\} \quad . \tag{2}$$

So, each function should be well defined for any arrangement of arguments that it may come across. Furthermore, the solution to the problem should be expressed by the combination of the procedures from the set of functions and the arguments from the set of terminals. For example, $\boldsymbol{\mathcal{F}} = \{\mathrm{AND}, \mathrm{NOT}\}$ is sufficient to express any Boolean function.

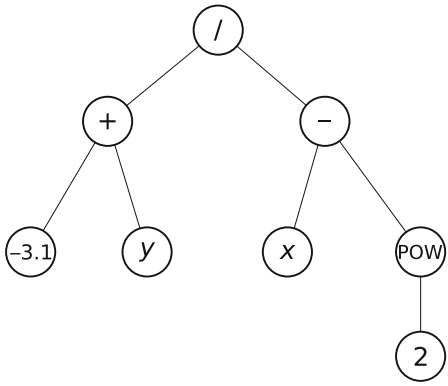

Fig. 1. An example of a parse tree as a chromosome of an genetic algorithm

The biological immune system has distributed elements as well as some features of artificial intelligence like an adaptation, learning, using memory, and associative retrieval of information in recognition [11]. Especially, the negative selection algorithm (NSA) can be applied for change detection because it uses the discrimination rule to classify some trespassers [10]. Detectors can be randomly generated to reduce those detectors that are not capable of recognizing themselves. However, detectors capable to distinguish intruders are kept to defense an organism. In the NSA, detection is performed probabilistically [3].

An antigen can support an antibody generation by stimulation a reaction against squatters. Besides, some positive viruses and bacteria cooperate with antigens [13]. An antibody (an immunoglobulin) is a large Y-shaped protein capable to recognize and deactivate external objects as negative bacteria and viruses [22]. It is worth to underline that the NSA can manage constraints in an evolutionary algorithm by dividing the population in two assemblies [6]. Antigens belong to the feasible solution sub-population, and "antibodies" – to the infeasible one.

The initial fitness for all antibodies in the current infeasible subpopulation is equal to zero. Next, a randomly selected antigen G^- from the feasible subpopulation is compared to the some chosen antibodies. After that, the match measure S between G^- and the antibody B^- is calculated due to the similarity at the genotype level. This measure of genotype similarity for the chromosome integer coding is, as follows [1]:

$$S(G^-, B^-) = \sum_{m=1}^{M} |G_m^- - B_m^-| \ , \tag{3}$$

where:
M – the length of the solution,
G_m^- - value of the antigen at position m, $m = \overline{1, M}$,
B_m^- - value of the antibody at position m, $m = \overline{1, M}$.

The negative selection can be modeled by an evolutionary algorithm, which prefers infeasible solutions that are similar to randomly chosen feasible one in the current population. We assume that all random choices of antigens are based on the uniform distribution.

The situation is different in the case of antibodies. If the fitness of the selected winner is increased by adding the amount of the similarity measure, then an antibody may pass over because of the relatively small value of assessment (3). On the other hand, some constraints may be satisfied by this alternative. What is more, if a constraint is exceeded and the others are not, the value of a similarity measure may be lower for some cases. One of two similar solutions, in genotype sense, may not satisfy this constraint and another may satisfy it.

4 An Extended NSA*

To avoid above disadvantages, some similarity measures can be developed from the state of an antibody B^- to the state of the selected antigen G^-, as below:

$$f_n(B^-, G^-) = \begin{cases} g_k(B^-) - g_k(G^-), k = \overline{1,K}, n = k, \\ |h_l(B^-)|, l = \overline{1,L}, n = K + l, \end{cases} n = \overline{1,N}, N = K + L \quad (4)$$

where
$g_k(x) \leq 0, k = \overline{1,K}$,
$h_l(x) = 0, l = \overline{1,L}$.

The distance $f_n(B^-, G^-)$ between B^- and G^- is supposed to be minimized for all constraint indexes n. If $f_n(B^-, G^-) < f_n(C^-, G^-)$, then B^- ought to be preferred to C^- due to the nth constraint. Moreover, if B^- is characterized by all shorter distances to the antigen than the antibody C^-, then B^- should be preferred for all constraints. However, some situations may occur when B^- is characterized by the shorter distances for some constraints and C^- is marked by the shorter distances for the others. In this case, it is difficult to select an antibody. So, a ranking procedure can be applied to calculate fitness of antibodies and then to select the winner.

In a ranking procedure, distances between the chosen antigen and some antibodies are calculated due to their ranks [2]. If B^- is characterized by the rank $r(B^-)$ such that $1 \leq r(B^-) \leq r_{\max}$, then the increment of the fitness function is estimated, as below:

$$\Delta f(B^-) = r_{\max} - r(B^-) + 1 \ . \quad (5)$$

Subsequently, some fitness values of selected antibodies are increased by their given increments. Then antibodies are returned to the current population and this process is repeated typically three times the number of antibodies. Each time, a randomly chosen antigen is compared to the same subset of antibodies.

Afterwards, a new population is constructed by selection, crossover and mutation without calculations of fitness. That process is repeated until a convergence of population emerges or until a maximal number of iterations is exceeded. At the end, the final population as outcomes from the negative selection algorithm is re-turned to the external evolutionary algorithm.

5 Optimization Model for Volunteer Grid

In the grid and volunteer computing systems like BOINC or Comcute, some scientific projects are transformed to a set of the calculation tasks that are executed concurrently by volunteer computers with a support of some levels of the middle-ware modules. A society of scientists can use these systems for extensive distributed calculations in some research projects. The 24-hour average performance of the most popular volunteer system BOINC is 8.186 TeraFLOPS. Moreover, the number of active volunteers can be estimated as 238,412, and also 388,929 computers process data [4].

In the Comcute system, an application for the Collatz hypothesis verification and another one for finding the 49th Mersenne number were applied to prove the intense human interactions, scalability and high performance [7].

In the architecture of the volunteer grid Comcute (Fig. 2), we can distinguish the Z-layer where the system client defines new tasks, starts instances of previously defined tasks, tracks statuses of running tasks and fetches results for completed tasks. On the other hand, the W-server layer supervises execution of tasks. For each task instance, a subset of W-servers is arranged that partitions the task among its members. The tasks pass input data packets for the task instance to connected S-servers beneath them as well as collect and merge results obtained from the S-layer. S-server is a distribution server that is exposed to clients who fetch execution code and subsequent data packets and return results for these data packets. I-client level is an untrusted layer of volunteers fetching and returning results to the system.

To test the ability of the MGP with NSA* for handling constraints, we consider a multi-criterion optimisation problem for task assignment in a distributed computer system [2]. Especially, MGP can minimize $Z_{\max}$ – the workload of a bottleneck computer and C – the cost of machines, concurrently.

A set of parallel tasks $\{T_1, \ldots, T_v, \ldots, T_V\}$ communicated with each other is considered among the coherent computer network with hosts located at the processing nodes from the given set $W = \{w_1, \ldots, w_i, \ldots, w_I\}$. Let the task T_v be executed on some hosts taken from the set of available sorts $\Pi = \{\pi_1, \ldots, \pi_j, \ldots, \pi_J\}$. The over-head execution time of the task T_v by the computer π_j is represented by an item t_{vj}.

The first criterion is a total host cost, as follows:

$$C(x) = \sum_{i=1}^{I} \sum_{j=1}^{J} \kappa_j x_{ij}^{\pi} \tag{6}$$

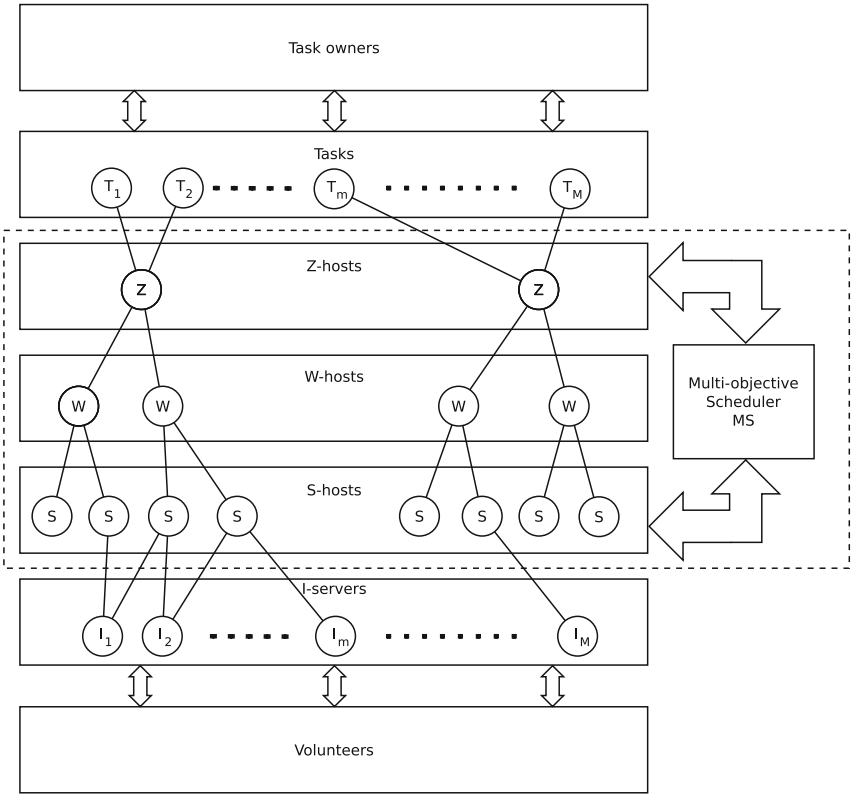

Fig. 2. Architecture of the Comcute system

where
$x = \left[x_{11}^m, \ldots, x_{vi}^m, \ldots, x_{VI}^m, x_{11}^\pi, \ldots, x_{ij}^\pi, \ldots, x_{IJ}^\pi\right]^T$,

$$x_{ij}^\pi = \begin{cases} 1 & \text{if } \pi_j \text{ is assigned to the } w_i, \\ 0 & \text{otherwise,} \end{cases}$$

$$x_{vi}^m = \begin{cases} 1 & \text{if task } T_v \text{ is assigned to the } w_i, \\ 0 & \text{otherwise,} \end{cases}$$

κ_j – the cost of the host π_j.

Another criterion is $Z_{\max}$ – a workload of the bottleneck host that is supposed to be minimized. It is provided by the subsequent formula:

$$Z_{\max}(x) = \max_{i \in \overline{1,I}} \left\{ \sum_{j=1}^{J} \sum_{v=1}^{V} t_{vj} x_{vi}^m x_{ij}^\pi + \sum_{v=1}^{V} \sum_{\substack{u=1 \\ u \neq v}}^{V} \sum_{i=1}^{I} \sum_{\substack{k=1 \\ k \neq i}}^{I} \tau_{vuik} x_{vi}^m x_{uk}^m \right\} , \qquad (7)$$

where
τ_{vuik} – the total communication time between the task T_v assigned to the ith node and the T_u assigned to the kth node.

Fig. 3 shows the workload of the bottleneck computer for the instance with 15 modules and two hosts. There are 30 decision variables and 7.394 admissible module assignments. An optimal workload of the bottleneck host is 47 [TU] versus the maximal one 102 [TU]. Even a small movement of a task to another host or a substitution of host sort can cause a relatively big alteration of its workload. What is more, there are two optimal solutions, as follows:
x*(1)=[1, 1, 1, 1, 1, 1, 1, 2, 2, 1, 2, 2, 2, 2, 2]
x*(2)=[2, 2, 1, 1, 2, 2, 2, 2, 2, 1, 1, 1, 1, 1, 1]

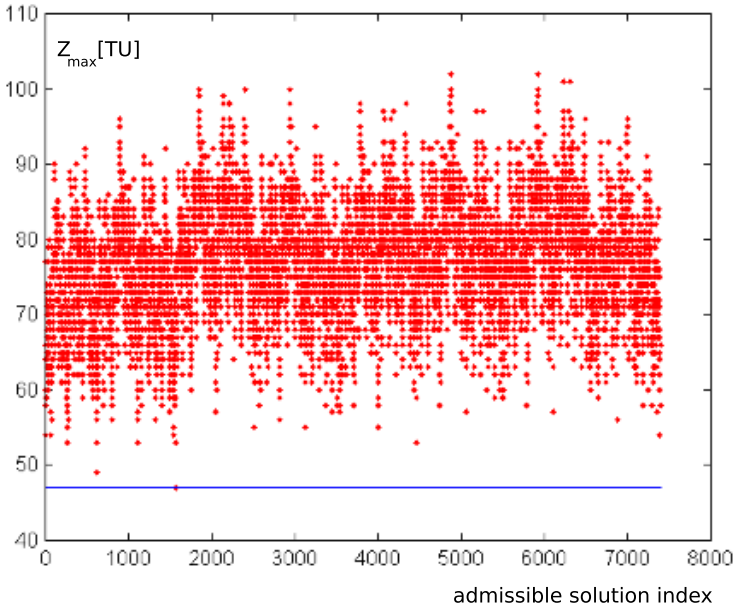

Fig. 3. Workload of the bottleneck computer for generated solutions

A host is supposed to be equipped with necessary capacities of resources. Let the memories $z_1, \ldots, z_r, \ldots, z_R$ be available in the volunteer system and let d_{jr} be the capacity of memory z_r in the host π_j. We assume the task Tv holds c_{vr} units of memory z_r during a program execution. The host memory limit cannot be exceeded in the ith node, as bellow:

$$\sum_{v=1}^{V} c_{vr} x_{vi}^{m} \leq \sum_{j=1}^{J} d_{jr} x_{ij}^{\pi}, i = \overline{1, I}, r = \overline{1, R} \ . \tag{8}$$

Let π_j be distributed independently according to the exponential distribution with rate λ_j. Hosts and tasks like Z, W or S can be allocated to nodes to guarantee the required reliability R, as below [1]:

$$\prod_{v=1}^{V}\prod_{i=1}^{I}\prod_{j=1}^{J}\exp\left(-\lambda_j t_{vj} x_{vi}^{m} x_{ij}^{\pi}\right) \le R_{\min} \ . \tag{9}$$

Let $(\boldsymbol{\mathcal{X}}, F, P)$ be the multi-criterion optimization question for finding the representation of Pareto-optimal solutions [6]. It can be established, as follows:

1. $\boldsymbol{\mathcal{X}}$ - an admissible solution set

$$\boldsymbol{\mathcal{X}} = \{x \in \boldsymbol{\mathcal{B}}^{I(V+J)} \mid \sum_{v=1}^{V} c_{vr} x_{vi}^{m} \le \sum_{j=1}^{J} d_{jr} x_{ij}^{\pi}, i = \overline{1,I}, r = \overline{1,R};$$
$$\prod_{v=1}^{V}\prod_{i=1}^{I}\prod_{j=1}^{J}\exp\left(-\lambda_j t_{vj} x_{vi}^{m} x_{ij}^{\pi}\right) \le R_{\min}; \sum_{i=1}^{I} x_{vi}^{m} = 1, v = \overline{1,V};$$
$$\sum_{j=1}^{J} x_{ij}^{\pi} = 1, i \ = \overline{1,I}\}$$

   where: $\boldsymbol{\mathcal{B}} = \{0, 1\}$,
2. F - a vector quality criterion

$$F : \boldsymbol{\mathcal{X}} \to \boldsymbol{\mathcal{R}}^2 \tag{10}$$

   where:
   $\boldsymbol{\mathcal{R}}$ – the set of real numbers,
   $F(x) = [Z_{\max}(x), C(x)]^T$ for $x \in \boldsymbol{\mathcal{X}}$,
   $Z_{\max}(x)$ and $C(x)$ are calculated by (7) and (6) respectively.
3. P - the Pareto relation [8].

To solve this problem we can apply the Strength Pareto Evolutionary Algorithm SPEA [24] or the Adaptive Multi-Criterion Evolutionary Algorithm with Tabu Mutation AMEA+ [1]. Moreover, some scheduling algorithms based on tabu search studied in [21] can be combined with an evolutionary approach.

In AMEA+, a tabu search procedure was applied as the second mutation operator to decrease the workload of the bottleneck computer. Moreover, we introduced the NSA* to improve the quality of obtained solutions and the evolutionary algorithm was denoted as AMEA*.

6 Numerical Experiments

For the instance with 15 tasks, 4 nodes, and 5 computer sorts, there are 80 binary decision variables. An average level of convergence to the Pareto set is 17.7% for the MGP* and 17.4% for the AMEA*. A maximal level is 28.5% for the MGP*

and 29.6% for the AMEA*. For this instance the average number of optimal solutions is 19.5% for the MGP* and 21.1% for the AMEA*. Fig. 4 6 shows the process of finding efficient task assignment by MGP* for the cut obtained from the evaluation space according to the cost criterion C and the workload of the bottleneck computer $Z_{\max}$. An average level of convergence to the Pareto set, an maximal level, and the average number of optimal solutions become worse, when the number of task, number of nodes, and number of computer types increase. An average level is 37.7% for the MGP* versus 35,7% for the AMEA*, if the instance includes 50 tasks, 4 nodes, 5 computer types and also 220 binary decision variables.

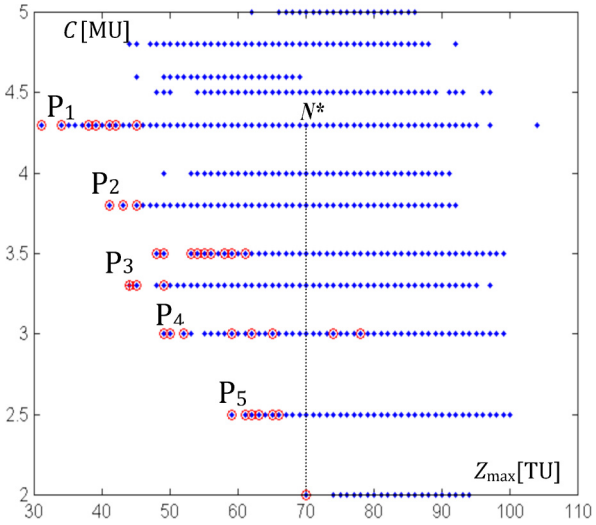

Fig. 4. Pareto front determined by GMP*

Concluding Remarks

Multi-objective genetic programming is relatively new paradigm of artificial intelligence that can be used for finding Pareto-optimal solutions. A computer program as a chromosome gives possibility to represent knowledge that is specific to the problem in more intelligent way than the data structure.

Our future works will focus on testing the other sets of procedures and terminals to find the Pareto-optimal task assignments for different criteria and constraints. Initial numerical experiments confirmed that sub-optimal in Pareto sense task assignments can be found by genetic programming. That approach permits for obtaining comparable quality outcomes to advanced evolutionary algorithm. Us-ing volunteer model of computations based on our Comcute system we plan implement large scale text classifier [9]. This task will allow us to evaluate the proposed architecture for real life tasks. Also the implementation will served as a proof of concept of an easy integration model for distributed computational nodes.

References

1. Balicki, J.: Immune Systems in Multi-criterion Evolutionary Algorithm for Task Assignments in Distributed Computer System. In: Szczepaniak, P.S., Kacprzyk, J., Niewiadomski, A. (eds.) AWIC 2005. LNCS (LNAI), vol. 3528, pp. 51–56. Springer, Heidelberg (2005)
2. Balicki, J.: Multicriterion Genetic Programming for Trajectory Planning of Underwater Vehicle. Int. Journal of Computer Science and Network Security 6, 1–6 (2006)
3. Bernaschi, M., Castiglione, F., Succi, S.: A High Performance Simulator of the Immune System. Future Generation Computer System 15, 333–342 (2006)
4. BOINC. Open-source software for volunteer and grid computing, `http://boinc.berkeley.edu/` (accessed January 25, 2014)
5. Cao, L., Gorodetsky, V., Mitkas, P.A.: Agent Mining: The Synergy of Agents and Data Mining. IEEE Intelligent Systems 24(3), 64–72 (2009)
6. Coello Coello, C.A., Van Veldhuizen, D.A., Lamont, G.B.: Evolutionary Algorithms for Solving Multi-Objective Problems. Kluwer Academic Publishers, New York (2002)
7. Comcute. Volunteer grid at Gdask University of Technology, `http://comcute.eti.pg.gda.pl/` (accessed January 25, 2014)
8. Deb, K.: Multi-Objective Optimization using Evolutionary Algorithms. John Wiley & Sons, Chichester (2001)
9. Draszawka, K., Szymanski, J.: Thresholding strategies for large scale multi-label text classifier. In: Proceedings of Human System Interaction IEEE (2013)
10. Forrest, S., Perelson, A.S.: Genetic Algorithms and the Immune System. In: Schwefel, H.-P., Männer, R. (eds.) PPSN 1990. LNCS, vol. 496, pp. 320–325. Springer, Heidelberg (1991)
11. Jerne, N.K.: Idiotypic Networks and Other Preconceived Ideas. Immunological Revue 79, 5–25 (1984)
12. Khan, D.: CAKE Classifying, Associating and Knowledge DiscovEry - An Approach for Distributed Data Mining (DDM) Using PArallel Data Mining Agents (PADMAs). In: IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, vol. 3, pp. 596–601 (2008)
13. Kim, J., Bentley, P.J.: Immune Memory in the Dynamic Clonal Selection Algorithm. In: Proc. the First Int. Conference on Artificial Immune Systems, Canterbury, Australia, pp. 57–65 (2002)
14. Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. The MIT Press, Cambridge (1992)
15. Koza, J.R., Keane, M.A., Streeter, M.J., Mydlowec, W., Yu, J., Lanza, G.: Genetic programming IV. Routine Human-Competitive Machine Intelligence. Kluwer Academic Publishers, New York (2003)
16. Li, H.X., Chosler, R.: Application of Multilayered Multi-Agent Data Mining Architecture to Bank Domain. In: International Conference on Wireless Communications, Networking and Mobile Computing, WiCom 2007, pp. 6721–6724 (2007)
17. O'Leary, D.E.: Artificial Intelligence and Big Data. IEEE Intelligent Systems 28(2), 96–99 (2013)
18. Samuel, A.L.: Programming Computers to Play Games. Advances in Computers 1, 165–192 (1960)
19. Sheble, G.B., Britting, K.: Refined Genetic Algorithm Economic Dispatch Example. IEEE Transactions on Power Systems 10, 117–124 (1995)

20. Snijders, C., Matzat, U., Reips, U.D.: Big Data: Big gaps of knowledge in the field of Internet. International Journal of Internet Science 7, 1–5 (2012)
21. Weglarz, J., Nabrzyski, J., Schopf, J.: Grid Resource Management: State of the Art and Future Trends. Kluwer Academic Publishers, Boston (2003)
22. Wierzchon, S.T.: Immune-based Recommender System. In: Hryniewicz, O., Kacprzyk, J., Koronacki, J., Wierzchon, S.T. (eds.) Issues in Intelligent Systems. Paradigms, pp. 341–356. Exit, Warsaw (2005)
23. Zhou, D., Rao, W., Lv, F.A.: Multi-Agent Distributed Data Mining Model Based on Algorithm Analysis and Task Prediction. In: 2nd International Conference on Information Engineering and Computer Science, pp. 1–4 (2010)
24. Zitzler, E., Deb, K., Thiele, L.: Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. Evolutionary Computation 8, 173–195 (2000)