Volunteer Computing System Comcute with Smart Scheduler

JERZY BALICKI, TOMASZ BIELIŃSKI, WALDEMAR KORŁUB, JACEK PALUSZAK
Faculty of Electronics, Telecommunications and Informatics
Gdańsk University of Technology
G.Narutowicza 11/12 St., 81-230 Gdańsk
POLAND
balicki@eti.pg.gda.pl, tombieli@student.pg.gda.pl, stawrul@gmail.com, jacekpaluszak@gmail.com

Abstract: - In this paper, a volunteer grid called Comcute is studied. Moreover, the harmony search scheduler is proposed. This scheduler has been designed for efficient using some resources of volunteer grid. The harmony search scheduler optimizes both a workload of a bottleneck computer and the cost of grid. Finally, some experiment outcomes have been discussed.

Key-Words: - volunteer grid, smart scheduling, harmony search algorithm, multi-objective optimization

1 Introduction

The Comcute system has been developed at the Gdańsk University of Technology in 2012 [8]. It is the development of volunteer computing paradigm because it allows volunteers to define some tasks and start computations with using available computers via the web. Especially, an application is systematically run for the Collatz hypothesis verification as well as another one for finding the 49th Mersenne prime. These were applied to prove the intense human interactions, scalability and high performance. Volunteers can link to the Comcute to download a task code and subsequently input data packets. Then, computers of volunteers return results to the system servers [8].

In the volunteer computing systems VCSs like BOINC, some scientific project applications are transformed to a set of the calculation tasks that can be executed concurrently by a large number of volunteer computers [7]. A society of scientists uses VCSs for extensive distributed calculations in many research projects. The 24-hour average performance of the BOINC is 8.186 TeraFLOPS. Moreover, the number of active volunteers can be estimated as 238,412, and also 388,929 computers is linked [7].

Some schedulers play important role in volunteer grids [4]. They can balance workload of the grid computers to improve the efficiency of VCS. Moreover, we can optimize the other parameters of the grid like the probability that all tasks meet their deadlines, cost of system, or a reliability of the distributed system [5]. If some decision preferences are based of two or more criteria, the scheduler solve multi-objective optimization problem [9].

There are three approaches for multi-criteria scheduling in the Comcute system. The first one is based on an evolutionary algorithm that was detailed described in [3]. The second alternative scheduler develops genetic programming that was discussed in [2]. However, we propose applying the harmony search meta-heuristics to improve scheduling process.

In this paper, a volunteer grid Comcute is studied. Moreover, the harmony search scheduler is described as well as multi-criteria genetic programming approach. Schedulers have been designed for efficient using some resources of grid and volunteer computers. The scheduler optimizes both a workload of a bottleneck computer and the cost of grid. Finally, some numerical experiments have been presented.

2 VCS Comcute

The Comcute consists of five layers (Fig.1) [8]. One of the most important is the *W* layer that is a vital fragment of volunteer computing grid system. Its main task is management of the computing power provided by *I*-nodes. This distribution layer consists of *W*-nodes to support reliable and malfunction resistant computation process. Above two features are obtained by data replication [7].

At the beginning of computation task, the group of *W*-nodes is organized. This group is divided regarding two aspects. First aspect is computation parallelization that speedups the computing process. Each node in group receives his subspace of input data. Subspaces of input data are generated by partition algorithm suitable for input problem.

ISBN: 978-960-474-381-0

The second aspect is related to reliability and its parameter λ_1 is the number of the redundant *W*-nodes for processing the same fragment of input data. λ_1 is equal to 2 by default. When reliability mechanism is on, the results acquired by λ_1 *W*-nodes can be compared. In next phase, an outcome of inconsistency can be detected or even repaired.

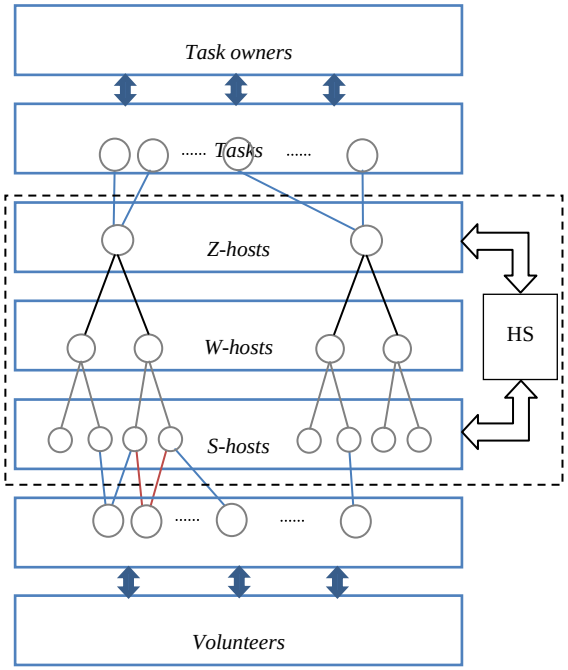

Fig.1. An architecture of the Comcute system

At the end of computation process, all results are synchronized regarding computation speed-up dimension in group. After that, the linker algorithm performs merging on each node. Merged results are presented by volunteer grid to the task owners.

Each *W*-node consists of a task managing module as well as a database module. Moreover, the *S*-communication module and the *W*-communication module are in node. We can indicate the *W*-layer monitoring module, too.

A task managing module handles all task related events like task creation, group gathering, input data spreading, results comparison and synchronization. Some database modules perform persistent storage operations like input data and results saving. The *S*-communication module is used to send computation order with data to *I*-nodes via *S*-nodes. Also results are received from *I*-nodes through this module.

The *W*-communication module is used for exchanging information with other *W*-nodes, especially input data replications and result synchronization. The *W*-layer monitoring module exchanges information about a node in entire layer. Load information from *W*-nodes is used to create optimal group for new task. This module allows detecting other node malfunction and performing suitable actions like overtaking damaged *W*-node input data subspaces.

Additionally, in the architecture of the volunteer and grid system Comcute (Fig. 1), 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. Some 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.

The *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.

The *I*-client level is consists of volunteer computers that download code and input data, and then they return results to the VCS. Volunteer computers communicate with the distribution layer and it is impossible to communicate with management layer or *Z*-layer.

3 Scheduling Model

A smart scheduler plays important role on the VCS like Comcute or BOINC regarding an efficiency improvement. We propose the harmony search for a scheduler support [1]. This approach will be compare to scheduler based on an evolutionary algorithm the other scheduler using genetic programming.

To test the ability of the harmony search scheduler HS2, we consider a multi-criterion optimisation problem for task assignment in a distributed computer system [2]. HS2 minimizes both *C* - the cost of machines and $Z_{\max}$ - a workload of a bottleneck computer to achieve load balance in VCS.

Let owner and system tasks $T_1,...,T_v,...,T_V$ be considered to perform by hosts located at nodes $w_1,...,w_i,...,w_I$. Let T_v be executed on some potential hosts $\pi_1,...,\pi_j,...,\pi_J$. The run time of T_v by π_j is represented by t_{vj}. A total host cost is, as follows:

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

where

$x = [x_{11}^{m},...,x_{vi}^{m},...,x_{VI}^{m}, x_{11}^{\pi},...,x_{ij}^{\pi},...,x_{IJ}^{\pi}]^T$,

ISBN: 978-960-474-381-0

$$x_{ij}^{\pi} = \begin{cases} 1 \text{ if } \pi_j \text{ is assigned to the } w_i, \\ 0 \text{ in the other case.} \end{cases}$$

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

κ_j - the cost of the host π_j.

Another criterion is $Z_{\max}$ that is provided by the formula, as below [3]:

$$Z_{\max}(x) = \max_{i \in \overline{1,I}} \left\{ \sum_{j=1}^{J} \sum_{v=1}^{V} t_{vj} x_{vi}^{m} x_{ij}^{\pi} + \sum_{\substack{v=1 \\ u \neq v}}^{V} \sum_{\substack{u=1 \\ k \neq i}}^{V} \sum_{\substack{i=1 \\ k \neq i}}^{I} \sum_{k=1}^{I} \tau_{vuik} x_{vi}^{m} x_{uk}^{m} \right\}, \quad (2)$$

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

An important constraint is related to memory capacities. Let d_{jr} be the capacity of memory z_r ($r = 1,2,\ldots R$) in the host π_j. Because T_v holds c_{vr} units of memory z_r during a program execution, the host memory limit cannot be exceeded in the ith node, what can be written, as bellows [4]:

$$\sum_{v=1}^{V} c_{vr} x_{vi}^{m} \leq \sum_{j=1}^{J} d_{jr} x_{ij}^{\pi}, \; i = \overline{1,I}, \; r = \overline{1,R}. \quad (3)$$

For instance with 15 modules and two hosts, there are 30 decision variables, 32 768 possible solution, and 7 394 admissible ones. So, a greedy algorithm for workload balance in larger systems is inefficient for development. Figure 2 shows the workload of the bottleneck computer for generated possible solutions by a systematic way.

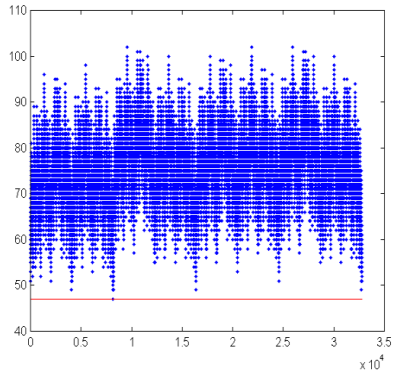

Fig.2. Evaluations of possible solutions for the instance with 15 tasks

Figure 3 shows admissible values of workloads for that instance. It has been observed that 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. An optimal workload of the bottleneck host is 47 [TU] versus the maximal one 102 [TU]. 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]

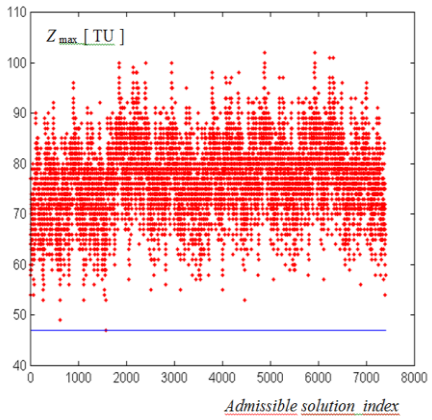

Fig.3. Admissible solution estimations for the instance with 15 modules

Let π_j be spoiled independently according to the exponential distribution with rate λ_j. We do not take into account of repair and recovery for a failed host. Instead, we shall allocate tasks to computers on which failures are least likely to occur during the execution of tasks. Hosts and system tasks can be allocated to nodes to guarantee the required reliability R, as below [3]:

$$\prod_{v=1}^{V} \prod_{i=1}^{I} \prod_{j=1}^{J} \exp(-\lambda_j t_{vj} x_{vi}^{m} x_{ij}^{\pi}) \leq R_{\min}. \quad (4)$$

4 Problem Formulation

HS2 solves the multi-criterion optimisation problem $(\mathcal{X}, F, P)$ to find the representation of Pareto-optimal solutions [9]. It is established, as follows:

---

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

$$\mathcal{X} = \{x \in \mathcal{B}^{I(V+J)} |$$
$$\sum_{v=1}^{V} c_{vr} x_{vi}^{m} \leq \sum_{j=1}^{J} d_{jr} x_{ij}^{\pi}, \quad i = \overline{1,I}, \; r = \overline{1,R};$$

$$\prod_{v=1}^{V} \prod_{i=1}^{I} \prod_{j=1}^{J} \exp(-\lambda_j t_{vj} x_{vi}^{m} x_{ij}^{\pi}) \leq R_{\min};$$

$$\sum_{i=1}^{I} x_{vi}^{m} = 1, v = \overline{1,V}; \; \sum_{j=1}^{J} x_{ij}^{\pi} = 1, i = \overline{1,I} \}$$

where $\mathcal{B} = \{0, 1\}$

ISBN: 978-960-474-381-0

2) F - a quality vector criterion

$$F: \mathcal{X} \to \mathcal{R}^2, \quad (5)$$

where

$\mathcal{R}$ – the set of real numbers,

$F(x) = [Z_{\max}(x), C(x)]^T$ for $x \in \mathcal{X}$,

3) P – the superior relation in $\mathcal{R}^2$ that leads to criteria minimization.

To solve the muli-objective optimization problem we can apply the Adaptive Multi-Criterion Evolutionary Algorithm with Tabu Mutation AMEA+ [4]. Moreover, we introduced the improved negative algorithm NSA* to improve the quality of obtained solutions for the evolutionary algorithm AMEA*. For integer constrained coding of chromosomes there were 12 decision variables in the test optimization problem. The binary search space consisted of 1.0737×10^9 solutions and included 25 600 admissible ones.

5 MGP* scheduler

Another scheduler for VCS is based on genetic programming because a computer program can model a solution by more intelligent way [29]. Genetic programming starts from a goal, set of functions and arguments, and then it creates an application [22]. This paradigm uses the principle of natural selection, crossover and mutation to obtain a population of programs. Multi-criterion genetic programming MGP* has been proposed to determine the Pareto-optimal solutions [2].

MGP* uses AMEA* that operates on the population of tree objects. These objects are similar to the parse tree that most parsers construct from a source code. A tree consists of branches and nodes: a root node, a branch one, and a leaf. The size of the 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 and terminal ones. A functional node represents the procedure randomly chosen from the set of functions, as below [2]:

$$\mathcal{F} = \{list, +, -, *, /\} \quad (6)$$

where *list* – the procedure that convert $I(V+J)$ real numbers called *activation levels* on $I(V+J)$ output binary numbers

$x_{11}^m, \ldots, x_{1I}^m, \ldots, x_{vi}^m, \ldots, x_{VI}^m, x_{11}^\pi, \ldots, x_{1J}^\pi, \ldots, x_{ij}^\pi, \ldots, x_{IJ}^\pi$.

Each function should be able to accept, as its arguments, any value and data type that may possible be returned by the other procedure [22]. 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:

$$\mathcal{T} = \{a_1, \ldots, a_m, \ldots, a_M\} \quad (7)$$

So, each function should be well defined and closed 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.

MGP* is supported by a negative selection algorithm NSA to handle constraints. NSA is applied for detection of any change by using the discrimination rule to classify some trespassers [10]. Detectors that are not capable of recognizing themselves are reduced, but detectors capable to distinguish intruders are kept. In the NSA, detection is performed probabilistically [6].

An antigen is a particle that stimulates a reaction against squatters and it can support an antibody generation. Also, some viruses and bacteria cooperate with antigens [19]. The antigen is recognized by the antibody (immunoglobulin) that is a huge *Y*- shaped protein capable to recognize and deactivate external objects as "negative" bacteria and viruses [30].

The NSA can manage constraints in an evolutionary algorithm by dividing the population in two assemblies [18]. "Antigens" belong to the feasible solution subpopulation, and "antibodies" – to the infeasible solution subpopulation.

At the beginning of the NSA run, the initial fitness for all antibodies in the current infeasible subpopulation is equal to zero. Then, a randomly chosen antigen G^- from the feasible subpopulation is compared to the some selected 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 between the antigen and the antibody depends on their representation. This assessment of similarity for the integer version is, as follows [5]:

$$S(G^-, B^-) = \sum_{m=1}^{M} \left| G_m^- - B_m^- \right|, \quad (8)$$

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 modelled by an adjusted evolutionary algorithm which prefers infeasible solutions that are similar to randomly chosen feasible one in the current population. Figure 4 shows the Pareto representation that was determined bv scheduler GMP*.

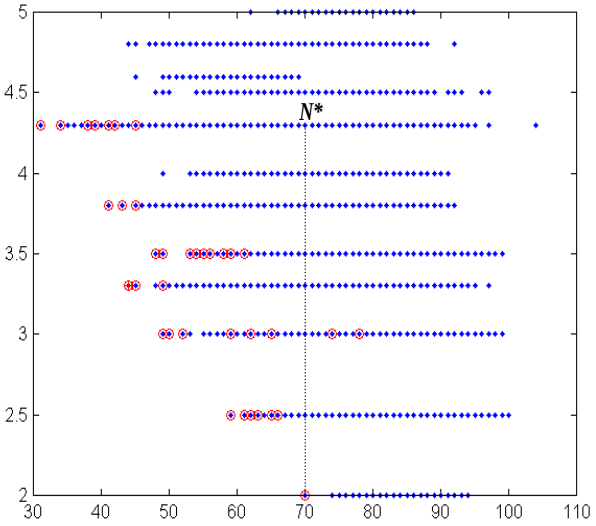

Fig.4. Pareto front determined by GMP*

6 Harmony search algorithm

A harmony search algorithm HS (HS) is a metaheuristic algorithm inspired by the improvisation process of musicians proposed by Zong Woo Geem in 2001 [11]. In this approach, there is modelled a band of musicians. Especially, some jazz musicians improvise and select appropriate melodic lines using their instruments to get the best sound while performing a piece of music [12].

We can compare a piece of music to an optimization problem, and a performing this music to a harmony algorithm. So, a musician can be modelled by a decision variable. Moreover, a music note can be interpreted as a value of decision variable. An action of playing a music note by a musician is reflected as generating any value of decision variables [14]. To conclude this comparison, a best harmony pf notes that are played by a band is a global optimum of the considered optimization problem [22]. Some musicians plays for finding a best harmony all together, but harmony search algorithm generates values of decision variables to find global optimum [13, 23].

A survey on applications of the harmony search algorithm has been submitted in [20, 24, 26]. A spectrum of the solved optimization problems by the harmony search is very wide, i.e. a layout of rectilinear branched networks [16] or water distribution networks [11]. Regarding water distribution networks design cost of them has been optimized [17]. Furthermore, vehicle routing [15] as well as a pipeline system [20] have been discussed. It is worth to mention that pump scheduling [21] and discrete structural optimization [25] can be made by the HS.

The first multi-criteria harmony search algorithm MHS has been proposed by Ricart, Hüttemann, Lima, and Barán in [28]. Then, two applications have been considered. The first one is the pump scheduling optimization [21], and the second one is related to an optimal design of groundwater remediation systems [25].

In this paper, we proposed an improved harmony search algorithm IHS, as follows.

1. The preparation of the input data to the optimization problem
2. The preparation of the objective function and decision variables.

$$\min f(x) \tag{9}$$

subject to $x_j \in X_j$, $j = 1, 2, ..., N$
where:
$f(x)$ – an objective function;
x – a decision variable vector , $x=[x_1,\ldots,x_j,\ldots,x_N]^T$;
x_j – the jth decision variable, $j= 1, 2, ..., N$;
N – a number of decision variables;
X_j – set of admissible values for x_j;

3. Determining the parameters of the harmonic algorithm
   - HMS – *Harmony Memory Size*;
   - HMCR – *Harmony Memory Considering Rate*, HMCR $\in$ [0,1];
   - PAR – *Pitch Adjusting Rate*, PAR $\in$ [0,1];
   - NG – *Number of Generations (Improvisations)*;
   - BW – *Bandwidth of Generations*, [-BW, BW];
4. Memory initialization.

HM stores initially random, then the best solution so far designated vectors and their corresponding values of the objective function (fitness).

Memory is a harmonic matrix algorithm (reminds population in genetic algorithm).

Initialization involves completing some randomly generated vectors of feasible solutions in terms of some constraints according to the formula:

$$x_i(j) = l(j) + \alpha(u(j) - l(j)) \tag{10}$$

where:
$i=1, 2, 3,..., HMS$ – an index for a vector of decision variables in the memory of the harmonic algorithm;
$j=1, 2, 3, ..., N$ – decision variable index;
α – a random number from the interval [0,1];
$u(j)$ – upper limit of the jth decision variable;
$l(j)$ – lower limit of the jth decision variable;

ISBN: 978-960-474-381-0

The memory of the algorithm is represented by a matrix:

$$HM = \begin{bmatrix} x_1^1 & x_2^1 & \dots & x_{N-1}^1 & x_N^1 \\ x_1^2 & x_2^2 & \dots & x_{N-1}^2 & x_N^2 \\ \dots & \dots & \dots & \dots & \dots \\ x_1^{HMS-1} & x_2^{HMS-1} & \dots & x_{N-1}^{HMS-1} & x_N^{HMS-1} \\ x_1^{HMS} & x_2^{HMS} & \dots & x_{N-1}^{HMS} & x_N^{HMS} \end{bmatrix} \begin{matrix} \rightarrow f(x^1) \\ \rightarrow f(x^2) \\ \vdots \\ \rightarrow f(x^{HMS-1}) \\ \rightarrow f(x^{HMS}) \end{matrix}$$

5. Generating a new solution (improvisation)

$$x'=(x'_1, x'_2, \dots x'_N).$$

Each variable is assigned a new value using the procedure, as follows:

Randomly generate r_1, r_2, r_3, r_4, r_5 from [0;1]
For j=1:N
If $r_1 \geq$ HMCR,
then
$x_i(j) = l(j) + r_2(u(j) - l(j))$
else
random selection for a vector index i
$x_i'(j)=x_{rand}$
if r_3<PAR
$x_i'(j) = x_i'(j) + (r_2 - r_5)bw|u(j) - l(j)|$
End

The algorithm generates new variables in two main ways:

- random generation with the probability (1-HMCR). A new value x'_i for x_i is randomly selected from X_i to ensure that $x'_i \in X_i$;
- selection from memory with the probability HMCR. The i-th variable x'_i, is randomly chosen regarding the roulette rule or the uniform distribution from a set $\{x^1_i, x^2_i, \dots, x^{HMS}_i\}$. In this case, the additional procedure is randomly made:
  - ✓ Tuning a new value x'_i that is modified by $(r_2 - r_5)bw|u(j) - l(j)|$. It permits finding a local optimum. This tuning is performer with probability PAR;

6. Analysis of obtained solutions.
When the solution is better than the worst harmony stored in the memory, it replaces the worst one. Solutions are compared by the objective function.
7 The stop criterion.
If the number of iterations NG is equal to the maximum NGmax, then stop. If not, back to step 4.

This modified harmony search algorithm can be adjusted for solving multi-objective optimization problems [27]. Our initial numerical experiments confirmed that proposed harmony search algorithm is capable to schedule tasks in the volunteer systems like Comcute.

7 Concluding remarks

In this paper, a volunteer grid Comcute is studied. Moreover, the harmony search scheduler is proposed. Scheduler has been designed for efficient using some resources of volunteer grid. The scheduler optimizes both a workload of a bottleneck computer and the cost of grid.

Our future works will focus on testing the other sets of criteria. Initial numerical experiments confirmed that sub-optimal in Pareto sense task assignments can be found by harmony search. That approach permits for obtaining comparable quality outcomes to multi-criteria genetic programming.

References:

[1] Ayob, M., Hadwan, M., Ahmad Nazr, M., and Ahmad, Z.: *Enhanced Harmony Search Algorithm for Nurse Rostering Problems.* Journal of Applied Sciences, 13(6), June 2013, 846-853.

[2] Balicki J., Balicka H., Masiejczyk J., Zacniewski A.: *Multi-criterion Decision Making in Distributed Systems by Quantum Evolutionary Algorithms*. Proceedings of The 2th European Conference on Computer Science, Puerto de la Cruz, Spain, November 30 – December 2, 2010, WSEAS Press, pp. 328-333.

[3] Balicki J.: *Multi-criterion Decision Making by Artificial Intelligence Techniques*. Proc. on the 8th Int. Con. on Artificial Intelligence, Knowledge Engineering and Data Bases, February 2009, Cambridge, pp. 319-324.

[4] Balicki J.: *Multi-criterion Tabu Programming for Pareto-optimal Task Assignment in Distributed Computer Systems, in: Mastorakis N.E., at el. "*New Aspects of Computers*", Crete 2008, WSEAS Press, pp. 142-152*

[5] Balicki J.: *An Adaptive Quantum-based Multi-objective Evolutionary Algorithm for Efficient Task Assignment in Distributed Systems,* Proc. of *The WSEAS Int. Conf. on Computers, July22-26, 2009, Rodos Island, Greece*, WSEAS Press, pp. 417-422

[6] Bernaschi M, Castiglione F, Succi S (2006) *A High Performance Simulator of the Immune System.* Future Generation Computer System, 15:333-342

[7] BOINC, http://www.boinc.com, Accessed 20 March 2014

[8] Comcute, http://comcute.eti.pg.gda.pl/, Accessed 20 March 2014

[9] Deb K (2001) *Multi-Objective Optimization using Evolutionary Algorithms*. John Wiley & Sons, Chichester

[10] Forrest S, Perelson AS (1991) *Genetic Algorithms and the Immune System.* Lecture Notes in Computer Science, pp. 320-325

[11] Geem, Z. and Cho, Y.: *Optimal Design of Water Distribution Networks Using Parameter-Setting-Free Harmony Search for Two Major Parameters.* Journal of Water Resources Planning and Management, 137(4), July 2011, pp. 377-380.

[12] Geem, Z. and Sim, K.: *Parameter-setting-free harmony search algorithm.* Applied Mathematics and Computation, 217(8), December 2010, pp. 3881-3889

[13] Geem, Z.: *Parameter Estimation of the Nonlinear Muskingum Model Using Parameter-Setting-Free Harmony Search.* Journal of Hydrologic Engineering, 16(8), August 2011, pp. 684-688.

[14] Geem, Z.W., Kim, J.H., Loganathan, G.V.: *A New Heuristic Optimization Algorithm: Harmony Search.* Simulation 76(2), 60–68 (2001)

[15] Geem, Z.W., Lee, K.S., Park, Y.: *Application of Harmony Search to Vehicle Routing.* American Journal of Applied Sciences 2(12), 1552–1557 (2005)

[16] Geem, Z.W., Park, Y.: *Harmony Search for Layout of Rectilinear Branched Networks.* WSEAS Transactions on Systems 5(6), 1349–1354 (2006)

[17] Geem, Z.W.: *Optimal Cost Design of Water Distribution Networks using Harmony Search.* Engineering Optimization 38(3), 259–280 (2006)

[18] Jerne NK (1984) *Idiotypic Networks and Other Preconceived Ideas*. Immunological Revue 79:5-25

[19] Kim J and Bentley PJ (2002) *Immune Memory in the Dynamic Clonal Selection Algorithm*. In: Proc the First Int Conference on Artificial Immune Systems, Canterbury, Australia, pp 57-65

[20] Kim, S.H., Yoo, W.S., Oh, K.J., Hwang, I.S., Oh, J.E.: *Transient Analysis and Leakage Detection Algorithm Using GA and HS Algorithm for a Pipeline System.* Journal of Mechanical Science and Technology 20(3), 426–434 (2006)

[21] Kougias, I. and Theodossiou, N.: *Multiobjective Pump Scheduling Optimization Using Harmony Search Algorithm (HSA) and Polyphonic HSA.* Water Resources Management, 27(5), March 2013, 1249-1261.

[22] Koza JR, Keane MA, Streeter MJ, Mydlowec W., Yu J, and Lanza G (2003) *Genetic programming IV. Routine Human-Competitive Machine Intelligence*. Kluwer Academic Publishers, New York

[23] Lee, K.S., Geem, Z.W., Lee, S.-H., Bae, K.-W.: *The Harmony Search Heuristic Algorithm for Discrete Structural Optimization.* Engineering Optimization 37(7), 663–684 (2005)

[24] Lee, K.S., Geem, Z.W.: *A New Structural Optimization Method Based on the Harmony Search Algorithm.* Computers and Structures 82(9-10), 781–798 (2004)

[25] Luo, Q., Wu, J., Sun, X., Yang, Y., and Wu, J.: *Optimal design of groundwater remediation systems using a multi-objective fast harmony search algorithm.* Hydrogeology Journal, 20(8), December 2012, pp. 1497-1510.

[26] Manjarres, D., Landa-Torres, I., Gil-Lopez, S., Del Ser, J., Bilbao, M., Salcedo-Sanz, S., and Geem, Z.: *A survey on applications of the harmony search algorithm.* Engineering Applications of Artificial Intelligence, 26(8), September 2013, 1818-1831.

[27] Paluszak J.: *Harmony search application in Java*, http://www.harmonysearch.info, Accessed 20 March 2014

[28] Ricart, J., Hüttemann, G., Lima, J., and Barán, B.: *Multiobjective Harmony Search Algorithm Proposals.* Electronic Notes in Theoretical Computer Science, 281, December 2011, 51-67.

[29] Samuel AL (1960) *Programming Computers to Play Games.* Advances in Computers 1:165-192

[30] Wierzchon ST (2005) *Immune-based Recommender System*. In: Hryniewicz O, Kacprzyk J, Koronacki J and Wierzchon ST (ed) Issues in Intelligent Systems. Paradigms. Exit, Warsaw, pp 341-356