Parallel packing procedure for three dimensional bin packing problem

Wojciech Bożejko
Department of Automatics,
Mechatronics and Control Systems,
Faculty of Electronics,
Wrocław University of Technology,
Janiszewskiego 11-17,
50-372 Wrocław, Poland
Email: wojciech.bozejko@pwr.edu.pl

Łukasz Kacprzak
Department of Automatics,
Mechatronics and Control Systems,
Faculty of Electronics,
Wrocław University of Technology,
Janiszewskiego 11-17,
50-372 Wrocław, Poland
Email: lukasz.kacprzak@pwr.edu.pl

Mieczysław Wodecki
Institute of Computer Science,
University of Wrocław,
Joliot-Curie 15,
50-383 Wrocław, Poland
Email: mwd@ii.uni.wroc.pl

***Abstract*—The paper proposes an algorithm for parallel packing procedure for three-dimensional packing problem. In the considered variant of the problem the load of the biggest capacity is placed inside the container with permanent dimensions. In order to parallelize computations there was CUDA technology used which enables running of independent computing threads with the use of graphic card cores. The obtained results prove the validity of the assumptions, particularly for large instances of the problem.**

I. Introduction

The problem of packing belongs to the group of optimization problems with wide practical application. Depending on the considered variant it is used for modeling such problems as transportation, storage, loading, the design of a very large scale integrated (VLSI) circuits, computer networks or paging. To put it simply, the problem of packaging can be reduced to the need of allocating of a certain number of elements to a number of containers, so as to satisfy the adopted criterion (or criteria). The discussed in the literature variants of the packing problem can be divided, for instance, with reference to the number of dimensions of the boxes (elements) or adopted constraints. Apart from the variant of the problem, for which one to three dimensions are being considered, the N-dimensional packing problem [1] is undoubtedly worth mentioning, wherein the time in which the element lies in the container is adopted as an additional dimension. In addition, the researchers pay special attention to variants of the problem, such as: single bin packing (loading into a single container of fixed dimensions), bin packing with variable height and strip bin packing (variable size of the container) or batch bin packing and multi-container loading (loading into many containers). Packing problem is related to a cutting problem where the process of loading items into the container is considered as a problem of dividing the container into smaller items.

The paper [2] describes a three-stage problem of two-dimensional packaging (3-stage 2BP). This problem is applied in the process of cutting of glass, steel or wood into smaller parts. Since industrial machines can often perform only movements determined at a given stage of work (vertical or horizontal), the authors are considering first only the ability to perform horizontal cuts, then vertical and horizontal again. In the paper there are not only mathematical models of the problem formulated but also there is a presentation of branch & price (B&P) algorithm.

Stawowy [2] developed an evolutionary algorithm for problems of one-dimensional packing. The aim of this study was to create a possibly simple algorithm for solving the problem at the level similar to the more specialized algorithms. In such a case the offspring is created by mutation operators, without the use of crossing.

The works: He et al. [4], Wu et al. [6], Karabulut and Mustafa [5] presented the problem of three-dimensional packing in which boxes are placed in the container of a fixed length and width but with a variable height. In the variant of the problem considered by He et al. [4] and Wu et al. [6] the boxes may have different sizes and rotations. Single solution is represented as a chromosome, containing information about the order of packing the boxes into the container and the kind of rotation of each of them. Goncalves and Resende [8] developed a parallel algorithm, wherein the chromosome comprises information concerning the order of packing boxes to container and the layer in which a given box can be placed. A layer is orthogonal structure (vertical or horizontal), consisting of boxes of the same type used to fill in the empty spaces inside the container (maximal spaces). The authors present the procedure for connecting the free space (called MaxJoin).

Bi-objective two-dimensional vector packing problem - Mo2-DBPP) was presented in [7]. In the work there were proposed two iterative approaches to solve the problem based on the properties of the Pareto frontier structure. The effectiveness of the two approaches has been tested using the test data literature.

In [9] there was considered the problem of three-dimensional bin packing (3DBPP) as a task maximizing two criteria: the total load volume and the number of cartons packed into a single container with fixed dimensions. The results of calculations obtained with the use of genetic algorithm and the simulated annealing algorithm were then compared.

In the paper [12] the two-dimensional irregular shape bin packing problem with guillotine constraints was introduced. The problem arises in the glass cutting industry. The authors presented a hybrid algorithm that is a constructive heuristic that determines the relative position of pieces in the bin and guillotine constraints via a mixed integer programme model.

978-1-4799-8701-6/15/$31.00 ©2015 IEEE.

To determine overlapping between the pieces, in the particular bin, NFP concept was implemented: given two convex polygons p_i and p_j, and a reference point on p_j, the NFP is formed by tracing the path of the reference point on p_j around the boundary of p_i in such a way that the polygons always touch and never overlap. If the reference point moves inside the NFP then p_i and p_j overlap. For computational experiments, different objective functions, initial permutations and approaches for associating a guillotine cuts were used. The constructive algorithm proposed obtained high quality results, improving the best known solutions in all benchmarks instances considered.

The implementation of bin packing problem for dynamic resource allocation strategies in virtualized data centers was proposed in work [13]. The virtualization technology allows to build multiple virtual servers on one physical host. That leads to many benefits, such as lower energy consumption, greater effectiveness and reliability. The authors discussed literature algorithms for static and dynamic resource allocation. The paper considered initial placement of available virtual machines, further migrations between the hosts and allocation of virtual servers created in time. For the experiments there was hardware infrastructure designed, with six identical hosts and ninety virtual machines.

The implementation of genetic algorithm for multi-criteria network scheduling problem was described in paper [14]. Authors considered computer system consisted of M back-end servers and a single machine acting as a task dispatcher. The tasks to be assigned to the servers were defined by three parameters: arrival, execution and deadline time. The online version of enhanced Local Search Elitist Non-dominated Sorting Genetic Algorithm for multi-criteria scheduling of incoming tasks was compared with other approaches to load-balancing (such as Round Robin, Shortest Job First and Earlies Deadline First implementations).

An extremely computationally expensive element of algorithms developed for packing problems is the packing procedure (also known as decoding procedure). This procedure relies most often in verifying the correctness of the solutions provided by the algorithm and evaluation of their quality. The purpose of the packing procedure for the variant of the problem under consideration in this work (see Section II) is to find the allowed positions inside the container for the sequence of boxes obtained by the simulated annealing algorithm. This process was parallellized with the use of independently working threads, activated by the graphic card cores. This procedure is applied to reduce the computatinal time of the whole algorithm and consequently enabling the performance of calculations for larger instances of the problem in a satisfactory time. The packing procedure is described in Chapter III. The results of calculations were presented in Chapter IV.

II. Problem description

The three-dimensional bin packing problem (3D-BPP) in the considered variant relies in packing of such number of boxes to a single container that, under the fulfillment of all the adopted constraints, the total volume of the load was the greatest. Container K (Figure 1) has a form of a cuboid with fixed dimensions: length L, width W, height H and volume V. There is not only a set of boxes given

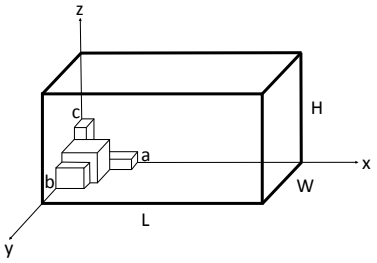

Fig. 1. Container

$P = \{p_0, \ldots, p_{n-1}\}$ but also types $T = \{t_0, \ldots, t_{m-1}\}$and rotation $R = \{0, 1, 2, 3, 4, 5\}$. Every box from a set P has specific type defining its dimensions and possible rotation, i.e. $p_i = t_k$, for $k \in T, i = 0, \ldots, n-1$. A given type can be saved in the form of: $t_k = (l_k, w_k, h_k, MR_k)$, where $MR_k \subseteq R$ determines the available rotations, whereas l_k, w_k, h_k denote respectively: the length, width and height of the box.

The boxes must be packed into the container K wherein they make up a set $PZ \subseteq P$, so that the adopted constraints are met and the sum of the volume of individual boxes was maximally:

$$\max \sum_{i=0}^{n-1} G_i, \tag{1}$$

where $G_i = c_i v_i$, $c_i = 1$ when the box is packed, $c_i = 0$ otherwise, $v_i = l_i w_i h_i$ is the volume of a single box. For determining the position of the boxes there were Cartesian coordinates in the reference system used. The following constraints have been adopted. Every packed box has to be fully placed inside the container, parallel to the side walls in one of the available for the type rotations. In the set MR_k there may be from one to six rotations. The box cannot occupy the space previously occupied by the packed boxes and needs to be placed on the bottom of the container or on the the upper part of another box.

The set of boxes P can be of any nature, starting from weakly heterogeneous up to strongly heterogeneous. If the set is weakly heterogeneous, this means that the instance of the problem contains few types, with plenty of boxes for each of them. The set of strongly heterogeneous consists of many types of boxes, with small amounts of them for each type.

III. Parallel packing procedure

A. Solution representation

The solution is represented as a chromosome, consisting of two parts (see Figure 2). The first of them (Boxes) represents the order received by procedure of packing boxes, while the other (Rotations) holds the number of rotations, which has been assigned to a given box. The box with the specified number, situated in the i-th field of the first part of the chromosome corresponds to the rotation number saved in the i-th field of the second part of the chromosome.

Both parts of the chromosome are equipped with the length (number of fields) equal to the number of boxes which means that for the number of boxes equal to n, the chromosome has 2n fields. The method of rotation of the boxes cannot be prohibited by the type to which it belongs.

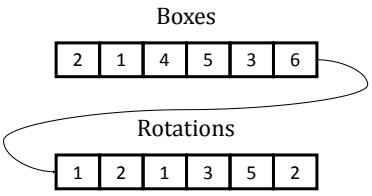

Fig. 2. Solution representation

B. The procedure

The purpose of the packing procedure is to evaluate the accuracy and quality of the solution encoded with the use of chromosome. Its basic elements are:

- chromosome which is the encoded representation of the considered solution,
- list of items containing the coordinates in the Cartesian reference system on which it is potentially possible to place a single box,
- container of a defined length, height and width,
- list of packed boxes, storing information concerning dimensions and positions of the boxes previously placed in the container.

Procedure considers the possibility to insert one box into the container, based on the sequence and rotation recorded in the representation of solutions. In each iteration one box with the number stored in the next chromosome field is taken into account. To allow insertion of the box into the container, it is necessary to find a position which is available in the list of positions, where the adopted restrictions will not be affected. For considered box, in a given iteration, there can exist more than one position for which no constraints are violated. Since the process of the analysis of a set of available positions, in the context of adopted constraints, is computationally expensive, it takes place with the implementation of parallel threads, using graphics card cores. The idea is illustrated in Figure 3, with the assumption that the list of items has five possible positions where the insertion of the box is possible. To each thread there

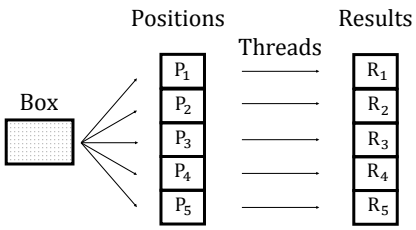

Fig. 3. Parallel computation

is a box assigned, which is currently under consideration, and one available position. As a result there is a table obtained containing the information whether the item meets accepted constraints and concerning the quality of its partial packing. The quality of partial packing is estimated according to the approach described in [4][10]. Next, the coefficient, which is the ratio of the sum of the volume of boxes, which create partial-packing, and the volume of the smaller container (ang. cage bin) is calculated:

$$I = \frac{V_l}{L_c \times W_c \times H_c}. \tag{2}$$

The smaller container is a container whose dimensions are determined by the largest coordinate of length, height and width of the load previously packed into it. Figure 1 shows a cuboid space, formed by connection of lines perpendicular to the axis of coordinates, outgoing points a, b, and c. Final position, for a given box, is selected among those satisfying constraints position, which has the highest value factor. This approach is applied to ensure consistency of the load placed in the container. The selected item is removed from the list of items and in its place there are three new created. Assuming that the box is inserted at position (x, y, z), the new positions have the following coordinates: (x + d, y + s, z + w), where d is the length of the box, s denotes its width, and w - its height. In the last step there is the total volume of the load placed in the container calculated. The value is identified with chromosome and is not more computed.

C. Simulated Annealing

In order to conduct the packing procedure there was simulated annealing algorithm chosen. The task of the algorithm is to provide new, coded solutions (chromosomes) which enable efficient search of solutions space. Simulated annealing algorithm can be placed among the local search methods. It was inspired by the annealing process which takes place in metallurgy. In the process it was noted that the cooled metal particles gradually form more regular structures. The important elements of the algorithm are: initial temperature, final temperature, probability function, in which worse solutions are accepted and cooling function.

In each iteration of simulated annealing algorithm, the neighborhood of solutions currently under consideration is tested. The neighborhood is constituted by solutions formed by small modifications of the current solution. If the neighborhood finds the solution qualitatively better than the current one, it replaces it. According to the probability function the current solution may be as well replaced by qualitatively worse solution. This system is designed not only to prevent stagnation but also to direct the process of searching in new areas of space solutions. The selection of temperature values, cooling scheme and the probability function affects the frequency of accepting solutions worse than the current one. In the implemented algorithm the neighborhood is created in two ways: by switching places of selected fields of chromosome and therefore the change in the sequence in which the packing procedure reads the successive boxes and by recording fields of the chromosome in reverse order.

IV. Computational experiments

The calculations were performed on the machine equipped with Intel Core i7 CPU X980 (3.33GHz) procesor and the

Linux operating system Ubuntu 12.04.5 LTS. Parallel computations were performed with the use of the Tesla K20 graphic card, having 5GB of memory (GDDR5) and 2496 cores (CUDA).

To generate test data there were instances of the problem

TABLE I. COMPUTATION TIME FOR ALGORITHM WITH PARALLEL PACKING PROCEDURE

ANB	3	5	8	10	12	15	20
41	0.17	0.10	0.10	0.08	0.09	0.10	0.09
45	0.18	0.11	0.10	0.10	0.09	0.09	0.10
50	0.18	0.11	0.10	0.10	0.10	0.09	0.10
56	0.18	0.11	0.12	0.09	0.11	0.09	0.10
63	0.20	0.10	0.09	0.09	0.10	0.11	0.10
71	0.23	0.15	0.15	0.10	0.11	0.09	0.12
80	0.23	0.13	0.17	0.09	0.13	0.08	0.11
91	0.26	0.17	0.15	0.10	0.12	0.10	0.13
104	0.29	0.26	0.18	0.10	0.13	0.14	0.14
120	0.31	0.28	0.20	0.11	0.15	0.15	0.15
138	0.36	0.25	0.23	0.12	0.17	0.13	0.17
161	0.42	0.29	0.22	0.13	0.20	0.16	0.12
190	0.52	0.47	0.31	0.15	0.21	0.22	0.19
225	0.67	0.53	0.39	0.18	0.28	0.30	0.29
270	0.79	0.64	0.52	0.20	0.36	0.14	0.35
328	1.08	0.82	0.69	0.27	0.45	0.50	0.46
404	1.43	1.03	0.88	0.31	0.62	0.58	0.57
504	1.85	1.72	1.19	0.41	0.78	0.74	0.87
641	2.40	2.29	1.65	0.55	1.12	1.10	1.20
832	3.59	2.91	2.27	1.25	1.63	1.58	1.74
1107	5.54	4.47	3.44	1.16	2.26	2.39	2.40
5127	76.21	78.98	50.89	12.63	26.96	28.76	30.81

described in [11] used. These data contain from three to twenty types of boxes, defining their dimensions, possible rotations and the dimensions of the container. With the increase the number of types the number of boxes for each of them is reduced. The initial test data were scaled so that it was possible to obtain instances of the problem with more differentiated number of boxes, while maintaining the same volume of the whole load. With respect to the fact that the increasing or reducing of the number of boxes has an impact on the number of operations performed by the packing procedure, it was possible to examine the time of the procedure, depending on the size of the instance.

Table I and Table II depict mean computation time, obtained

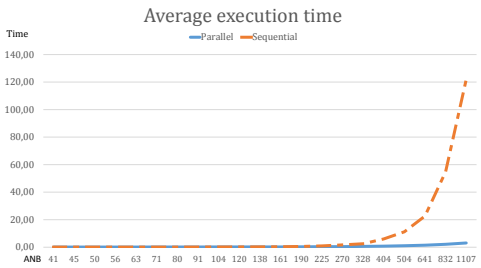

Fig. 4. Average execution time

for all startups of algorithm for data of specific number of types (3 to 20) and the number of boxes (ANB), expressed in seconds. For comparative purposes the computations were carried out also sequentially, without the use of the graphics card. The sequential packing procedure examines the possibil-

TABLE II. COMPUTATION TIME FOR ALGORITHM WITH SEQUENTIAL PACKING PROCEDURE

ANB	3	5	8	10	12	15	20
41	0.01	0.08	0.07	0.07	0.07	0.07	0.07
45	0.01	0.07	0.07	0.07	0.07	0.07	0.07
50	0.02	0.07	0.07	0.07	0.07	0.07	0.07
56	0.03	0.08	0.08	0.07	0.07	0.07	0.07
63	0.04	0.09	0.08	0.07	0.08	0.07	0.07
71	0.05	0.10	0.09	0.07	0.08	0.08	0.07
80	0.05	0.09	0.12	0.07	0.08	0.08	0.09
91	0.09	0.12	0.12	0.07	0.09	0.07	0.08
104	0.17	0.31	0.16	0.08	0.11	0.10	0.11
120	0.29	0.34	0.18	0.08	0.12	0.14	0.10
138	0.52	0.46	0.30	0.08	0.11	0.13	0.12
161	0.60	0.65	0.39	0.10	0.19	0.14	0.19
190	1.25	1.01	0.62	0.10	0.25	0.18	0.32
225	1.88	2.31	1.08	0.13	0.43	0.39	0.36
270	3.46	4.30	2.14	0.16	0.65	0.58	0.62
328	4.87	6.28	3.20	0.20	0.83	0.93	1.35
404	13.27	15.32	6.69	0.49	2.07	2.14	2.10
504	23.42	28.01	14.11	0.83	3.68	3.89	4.41
641	44.33	58.38	28.61	1.88	7.57	8.64	9.01
832	124.13	130.38	64.01	4.31	18.82	19.15	22.49
1107	264.72	269.60	163.14	10.29	45.91	49.43	50.47
5127	29235	32291	20006	1275	5394	6529	6933

ity of insertion of a considered box in each position which is available on the list of positions, one by one, calculating the value ratio, if the insertion is possible. The time profit resulting from the procedure implementing sequential calculations is the lack of necessity to copy the data to graphics card memory. A single run time of the algorithm consists of: the creation of the first individual to carry out the calculations by the packing procedure, generating neighbourhood and carrying out computations by packing procedure for the generated neighbourhood.

For smaller instances of the problem (up to a hundred boxes) better computation time was obtained with the use of the sequential packing procedure. For the boxes within a range of one hundred to one hundred and sixty the computation time was comparable. For data of above one hundred and ninety boxes the reported computation time was shorter using the parallel procedure. The obtained profit in time using the algorithm with the parallel procedure increases with the number of boxes, regardless of the number of types. Figure 4 shows the growth of mean computation time for all types of data, for which calculations were carried out parallely (Parallel, solid lines) and sequentially (Sequential, dashed line). Based on the data from Table I and Table II the speedup was calculated. The speedup is the ratio of running time of the algorithm with the sequential procedure to running time of the algorithm using a parallel procedure (Table III).

The value of the speedup increases with the size of the problem instance, regardless of number of types (Figure 5)

V. CONCLUSION

This paper presents an algorithm with a parallel packing procedure for three-dimensional packing problem. Parallel computations were performed with the use of CUDA technology. Parallelization of the process of checking the possibility of the insertion of a single box in a specific position gave

TABLE III. SPEEDUP

ABN	3	5	8	10	12	15	20
41	0.06	0.75	0.69	0.85	0.80	0.67	0.78
45	0.08	0.66	0.67	0.73	0.80	0.77	0.69
50	0.13	0.65	0.73	0.69	0.69	0.80	0.71
56	0.16	0.73	0.71	0.80	0.65	0.78	0.74
63	0.21	0.92	0.85	0.76	0.79	0.61	0.71
71	0.20	0.68	0.58	0.71	0.70	0.83	0.56
80	0.23	0.67	0.70	0.74	0.65	1.08	0.83
91	0.33	0.70	0.78	0.67	0.75	0.71	0.60
104	0.57	1.19	0.87	0.80	0.85	0.74	0.77
120	0.92	1.21	0.91	0.72	0.82	0.97	0.65
138	1.44	1.86	1.30	0.67	0.66	1.00	0.69
161	1.42	2.25	1.79	0.78	0.96	0.92	1.53
190	2.40	2.16	2.01	0.64	1.18	0.84	1.68
225	2.82	4.35	2.77	0.73	1.53	1.31	1.25
270	4.37	6.68	4.08	0.82	1.83	4.01	1.76
328	4.53	7.64	4.61	0.75	1.87	1.86	2.96
404	9.28	14.93	7.59	1.59	3.34	3.70	3.69
504	12.63	16.31	11.82	2.03	4.74	5.25	5.07
641	18.46	25.47	17.38	3.41	6.78	7.85	7.50
832	34.58	44.83	28.22	3.44	11.57	12.12	12.91
1107	47.77	60.26	47.40	8.90	20.30	20.66	21.06
5127	383.62	408.85	393.12	101.00	200.09	227.04	225.04

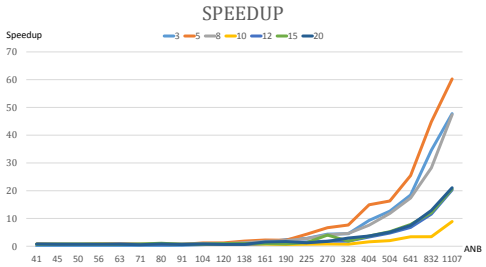

Fig. 5. Speedup

very promising results, for medium and large sizes of the input data. With reference to the fact that after placing the box in a single container there are three new positions addend to the list of potential items, the packing procedure in each iteration performs calculations for hundreds and even thousands of coordinates. Due to parallelization of computations, it was possibile not only to search of the whole set of available positions in a satisfactory time but also to choose the best position with reference to the adopted criterion. In connection to the fact that the packing procedure must be run on each new chromosome provided by the algorithm, the profit in time obtained due to the parallelization of computations allows for more efficient moves in the solution space.

REFERENCES

[1] H. Dyckhoff *A Typology of cutting and packing problems*, European Journal of Operational Research 44, 145-159, 1990

[2] J. Puchinger, G. R. Raidl *Models and algorithms for three-stage two-dimensional bin packing*, European Journal of Operational Research 183, 1304Ű1327, 2007

[3] A. Stawowy *Evolutionary based heuristic for bin packing problem*, Computers & Industrial Engineering 55, 465Ű474, 2008

[4] Y. He, Y. Wu, R. de Souza *A global search framework for practical three-dimensional packing with variable carton orientations*, Computers & Operations Research 39, 2395Ű2414, 2012

[5] K. Karabulut, M. M. Mustafa *A hybrid genetic algorithm for packing in 3D with deepest bottom left with fill method*, ADVIS 2004: Advances in Information Systems 2004, 441Ű450, 2004

[6] Y. Wu, W. Li, M. Goh, R. de Souza *Three-dimensional bin packing problem with variable bin height*, Computers & European Journal of Operational Research 202, 347Ű355, 2010

[7] J. F. Goncalves, M. G.C. Resende *A parallel multi-population biased random-key genetic algorithm for a container loading problem*, Computers & Operations Research 39, 179Ű190, 2012

[8] N. Dahmania, F. Clautiauxb, S. Krichena, E.-G. Talbib *Iterative approaches for solving a multi-objective 2-dimensional vector packing problem*, Computers & Industrial Engineering, Volume 66, Issue 1, pp. 158Ű170, 2013

[9] Ł. Kacprzak, J. Rudy, D. Żelazny *Multi-criteria 3-dimension bin packing problem*, Research in Logistics & Production. 2015, vol. 5, nr 1, s. 85-94, 2015

[10] TH Loh, AYC Nee *A packing algorithm for hexahedral boxes*, Proceedings of the conference of industrial automation. Singapore, 115 - 126, 1992

[11] E. E. Bischoff, M.S.W. Ratcliff *Issues in the Development of Approaches to Container Loading*, Omega, Int. J. Mgmt Sci. Vol. 23, No. 4, 377-390, 1995

[12] A. Martinez-Sykora, R. Alvarez-Valdes, J. Bennell, J. M. Tamarit *Constructive procedures to solve 2-dimensional bin packing problems with irregular pieces and guillotine cuts*, Omega 52, 15Ű32, 2015

[13] A. Wolke, B. Tsend-Ayush, C. Pfeiffer, M. Bichler *More than bin-packing: Dynamic resource allocation strategies in cloud data centers*, Information Systems 52, 83Ű95, 2015

[14] J. Rudy, D. Żelazny, *Memetic algorithm approach for multi-criteria network scheduling*, Proceeding of the International Conference on ICT Management for Global Competitiveness and Economic Growth in Emerging Economies (ICTM 2012), 247-261, 2012