Multi-Objective Pareto Optimization of Parallel Synthesis of Embedded Computer Systems

Mieczysław Drabowski
drabowski@pk.edu.pl
Cracow University of Technology, Kraków, Warszawska 24, 31-945, Poland

Summary
The paper presents problems of optimization of the synthesis of embedded systems, in particular Pareto optimization. The model of such a system for its design for high-level of abstract is based on the classic approach known from the theory of task scheduling, but it is significantly extended, among others, by the characteristics of tasks and resources as well as additional criteria of optimal system in scope structure and operation. The metaheuristic algorithm operating according to this model introduces a new approach to system synthesis, in which parallelism of task scheduling and resources partition is applied. An algorithm based on a genetic approach with simulated annealing and Boltzmann tournaments, avoids local minima and generates optimized solutions. Such a synthesis is based on the implementation of task scheduling, resources identification and partition, allocation of tasks and resources and ultimately on the optimization of the designed system in accordance with the optimization criteria regarding cost of implementation, execution speed of processes and energy consumption by the system during operation. This paper presents examples and results for multi-criteria optimization, based on calculations for specifying non-dominated solutions and indicating a subset of Pareto solutions in the space of all solutions.

Key words:
parallel synthesis, optimization, non-dominated solution, optimal Pareto set of solutions.

1. Model of computer embedded systems for their synthesis

Synthesis of complex and embedded systems is a multi-criteria optimization problem. The model of such a system for its high-level abstraction design and the algorithm realizing such a synthesis were presented in [1] and in [2], respectively. The starting point for constructing our approach to the issues of hardware and software synthesis is the deterministic theory of task scheduling [3], [4]. The theory may serve as a methodological basis for multiprocessor and multitasks system synthesis.
Accordingly, the decomposition of general task scheduling model was suggested, adequate to the problems of computer system synthesis. From the practical point of view such a model should examine the tasks, which may be either preemptable or non-preemptable. These characteristics are defined according to the scheduling theory. Tasks are preemptable when each task can be interrupted and restarted later without incurring additional costs. In such cases the schedules are called were preemptive. Similarly, if tasks cannot be interrupted, their schedules are non-preemptive.
Such a feature as preemptive of tasks in our approach cannot be a feature of the searched schedule – like occurs in the current model for scheduling tasks. The schedule applies to all the assigned tasks with individual attributes: preemptive, and non-preemptive. According to the existing system, the implementation of certain tasks must be no-preemptive, the other may be preemptive (which, in turn, influences significantly the selection of an appropriate scheduling algorithm), but on the other hand many system functions must be performed non-preemptive [5]. Moreover, we wish to specify the model of task scheduling in a way suitable for finding optimum control methods (in terms of certain criteria) – as well as optimum assignment of tasks – in terms of other criteria – all processors maybe universal (general) or specialized (dedicated). This is an essential change in relation to the approach in the allocation of tasks and resources in the system.
Thus, we were examines the system, which is of set consist three of subsets (Equation 1):

$$\textit{SYSTEM} = \{\textit{Resources, Tasks, Criteria}\} \quad (1)$$

Resources set (hardware and software) consists of P general processors $P = \{P_1, P_2,\ldots, P_p\}$ and the set of D additional, dedicated processors $D = \{D_1, D_2,\ldots, D_d\}$ [6].
Set of tasks consists of n tasks which are to be processed on a set of m processors and $m = p + d$. Each task is defined by a set of parameters: resource requirements, execution time, ready time and deadline, an attribute – preemptable or nonpreemptable. The set may contain defined precedence constraints represented by a digraph with nodes representing tasks, and directed edges representing precedence constraints. If there is at least one precedence constraint in a task set, we shall refer it to as a set of dependent tasks; otherwise we call it a set of independent tasks. The set of tasks form all the system functions, both outer applications and inner operating, diagnostic and also transmission processes. A feasible schedule is optimal, if its length is minimal. As for the optimality criteria for the system to be designed, we shall assume its maximum

Manuscript received March 5, 2021
Manuscript revised March 20, 2021
https://doi.org/10.22937/IJCSNS.2021.21.3.39

operating speed, minimum cost and minimum power consumption [7]. We will apply multi-criteria optimization in sense of Pareto. The solution is optimized in sense of Pareto if it is not possible to find a better solution, regarding at least one criterion without deterioration in accordance to other criteria. The solution dominates other ones if all its features are better. Pareto ranking of the solutions is the number of solutions in a pool which do not dominate it.

The process of synthesis will produce a certain number of non-dominated solutions. Although non-dominated solutions do not guarantee that they are an optimal Pareto set of solutions; nevertheless, in case of a set of suboptimal solutions, they constitute form of higher order optimal set in sense of Pareto and they give access to the shape of set of these solutions [8].

For example solution *W* can be improved both against criterion C_1 and C_2 – Fig. 1. For *P* and *Q* solutions, this possibility does not exist - an improvement on one criterion causes deterioration due to the second - they belong to the set of optimal solutions in the Pareto sense. Let's assume for example, that we want to optimize a solution of two contradictory requirements: the Cost and Power consumption – Fig. 2.

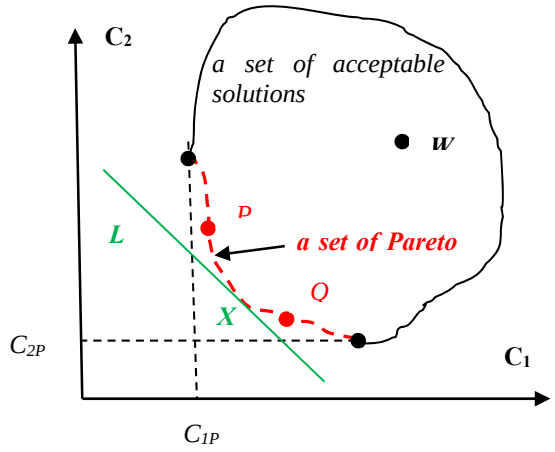

Fig. 1. Solutions for two criterions

It is possible to choose the wrong solution without knowing the shape of the curve of Pareto-optimal solutions (non-dominated solutions).

For example – Fig. 2.: The best solution is for Energy = 20 and Cost = 160. Designer - without knowledge of this curve - can choose the solution for Energy = 17 and Cost = 265), and yet there is a solution for Energy greater just by 1, and Cost less by as much as 100. Designer - without knowing the set of Pareto-optimal solutions - can also choose a solution with a Cost = 140 and with Energy = 29.

While using a traditional way with one optimization function, it is necessary to contain multi optimal criteria in one value. To do that, it is advisable to select properly the scales for the criteria; if the scales are selected wrongly, the obtained solution will not be optimal. The chart in the illustration shows where, using linearly weighed sum of criteria, we will receive the solution which may be optimizes in terms of all criteria.

The optimization of cost, power consumption and speed in the problem of synthesis is, undoubtedly, the problem where the potential number of solutions in sense of Pareto may be enormous. In order to bring multi-criteria optimization to a single criterion optimization one may use, for example, the method of weighted criteria; a substitute criterion equal to the sum of the weighted criteria (Equation 2):

$$MIN\ (X) = \Sigma\,(w_i \bullet C_i(X));\ \ where:\ 0 \le w_i \le 1,$$
where "i" is criterions number (2)

Then, the solution is the point of intersection of the set of permissible solutions with line, dependent on the values of weights of the interior criteria. To balance the impact of individual criteria, can make their normalization [10].

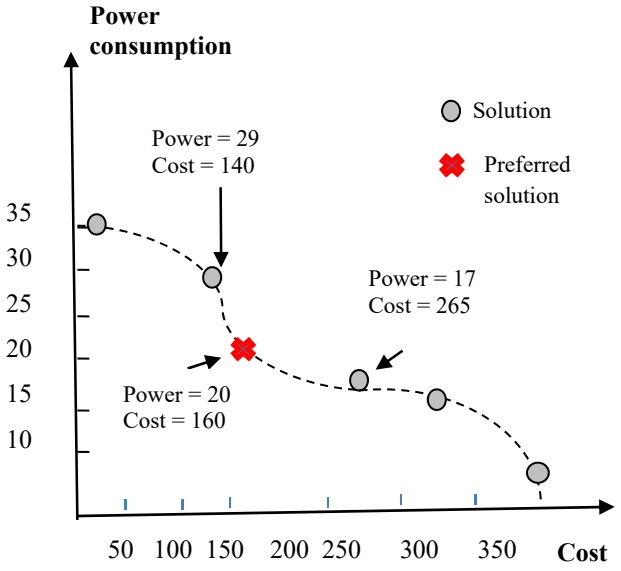

Fig. 2. The curve of optimal-Pareto solutions

Graphically, the solution can be presented as the point of intersection of the set of permissible solutions with line ***L*** (with the point ***X***), depending on the values of the criteria weights (Fig. 2. for two criterions). Due to the balanced impact of individual criteria, criteria may be standardized. The problem is to choose a priori values of criteria weights, which can lead to different solutions.

The suggested model may be used for defining various synthesis problems for optimum computer systems. The our model of a system in this approach, typical for the theory of task scheduling, consists of a set of requirements and existing relationships between them (related to their order,

required resources, time, readiness and completion deadlines, preemptive/non-preemptive, priority etc.). The synthesis procedure contains the following phases: identification of hardware and software resources for task implementation, defining the processing time, defining the conflict-free task schedule, defining the degree of concurrency in the performance, allocating the tasks for the recourses and indicating the operations which can be executed concurrent [11].

The synthesis has to perform the task partitioning into hardware and software resources. After performing the partition, the system shall be implemented partially by specialized hardware in the form of sub-assembles (most frequently integrated circuits) readily available in the resources pools or designed in accordance to the suggested characteristics. Software modules of the system are generated with the use of software engineering tools. Appropriate processors shall be taken from the resource consignment. Synthesis of a system may also provide a system operating control, create an interface and provide methods and components for synchronization and communication between the tasks implemented by software and hardware [12].

To sum up, the high-level synthesis of system, i.e. defining constraints and requirements of system, identifying its operations and resources, defining control should be implemented in synergy and be subject to multi-criteria optimization and verification during implementation.

The paper contains: presentation of a synergistic algorithm for the synthesis of embedded systems and its multi-criteria optimization in the sense of Pareto (Chapter 2) and the example of the implementation of this algorithm (Chapter 3).

2. Algorithm of parallel synthesis of embedded systems

Modeling the synergic – in our approach: of **parallel** – search for the optimum task schedule and resource partition of the designed system into hardware and software parts is fully justified. We suggest the following schematic diagram of a parallel process of synthesis computer systems [2] – Fig. 3. Simultaneous consideration of these problems may be useful in implementing optimum solutions, e.g. the cheapest hardware structures and shortest schedules. With such approach, the optimum task distribution is possible on the universal and specialized (dedicated) hardware and choice of resources with maximum efficiency.

The suggested parallel synthesis consists of: specification of requirements for the system to be designed and its interactions with the environment, defining all functions the system, search for the optimum task schedule and resource partition of system into hardware and software parts.

This synthesis consists in detail of the following steps:

1. Defining the tasks which fulfill the performance of system functions. Estimation of executing parameters comprised in the task procedures upon the available recourses (e.g. execution time or requirements for the memory space and defining the dependability of procedures) [13]. Defining constraints and critical requirements regarding accomplishment.

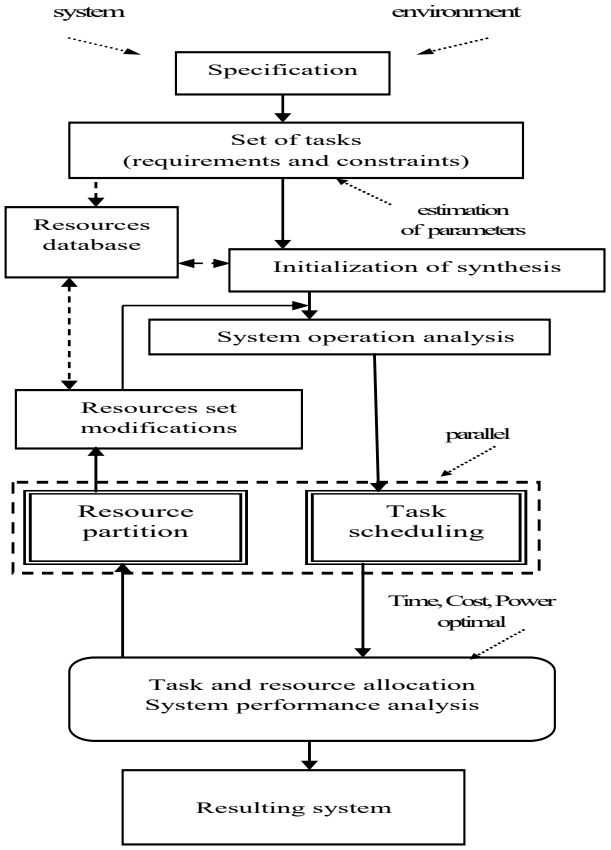

Fig. 3. Parallel synthesis of computer system

2. Assuming the initial values of resource set and task scheduling – the resource set and task schedule should be available (from the pool of resources and schedules and from the historical data base remembered due to the synthesis of systems similar to them in the past); i.e. they should meet all the requirements, though at this stage of the algorithm, maybe in a non-optimum way.
3. Task scheduling, resource partitioning and task and recourses allocation – all tasks of specification, resources currently selected and assigned to certain tasks.

4. Evaluating the operating speed, power consumption and system cost, etc., multi-objective optimization.
5. The every evaluation should be followed by a modification of the resource set and compute new schedule of tasks, as a result of a new system partitioning into hardware and software parts and the pursuit of achievement of a satisfying result.

Iterative calculations are executed till satisfactory design results are obtained – i.e. closer and closer to the optimal of system structure and schedule. The designed system should be fast, cheap, with low power consumption and dependable [14].

3. Results of multi-objective optimization for synthesis of computer embedded systems

We consider such example.
Tasks (system functions) used during the tests are generated as direction graphs and they are received as graphs STG http://www.kasahara.elec.waseda.ac.jp/schedule/
These generators have been worked out in order to standardize random tests for research into common task scheduling and allocating problems, especially for system synthesis and for such applications which need pseudo-random generating acyclic directed graphs. In generators, the sort of graph, number of source and sink nodes, the length of maximal track, the node and edge weight, degree of graph, probability of predecessors and successors' number etc. should be determined. For example, time of tasks might be generated as follows: the average time value for the task (e.g. = 5 units) and time of tasks determined by uniform distribution or regular distribution with a fixed standard deviation (e.g. = 1).
For the tests, maximum number of tasks have been determined = 100 (independent tasks) and 50 (dependent tasks), which is the sufficient number for the presentations of all algorithm features, and their comparisons as well (also to other algorithms) and is also the right number of operations for realistic system synthesis; obviously, system functions in its specification are given on the suitable level of granulation.
Resources applied in tests are shown in the following table (Table 1):
Tests were conducted into dependent, non-preemptive tasks. Parameters of constraints are: the maximum number of processors – 5, maximum cost – 3, maximum time – 25. Optimization criteria: cost, time and power consumption. As a result, a set of optimum solutions was received in sense of Pareto [15].
The following table shows a set of solutions in sense of Pareto, obtained as a result of algorithm performance for the problem: 15 dependent tasks, not considering the cost of operating memory.

Table 1. Resources applied in tests (ToP - Type of Processor, G-general processor, D-dedicated processor, MM-Memory module)

No.	ToP	ID	Speed	Cost	Power consumption
1	G	P1	1	1.00	0.01
2	G	P2	2	1.60	0.02
3	G	P5	5	2.20	0.05
4	G	P10	10	3.70	0.1
5	D	ASIC1	1	0.50	0.01
6	D	ASIC2	2	0.75	0.02
7	D	ASIC3	3	1.00	0.03
8	D	ASIC4	4	1.25	0.04
9	D	ASIC5	5	1.50	0.05
10	D	ASIC10	10	2.75	0.11
11	MM	PAO	1	0.2	0.001

The following table shows a set of solutions in sense of Pareto, obtained as a result of algorithm performance for the problem: 15 dependent tasks, not considering the cost of operating memory.

Table 2. Example solutions in sense of Pareto (NoS - Number of solution)

NoS	Cost	Time	Power consumption
1	2,75	4,2 *minimum*	77,69 *maximum*
2	1,5 *minimum*	8,4	54,42
3	2,95 *maximum*	18,5 *maximum*	20,9 *minimum*
4	2,55	19	23,9
5	2,95	15,5	24,01
6	2,75	18	22,96
7 *COM-*	**2,95 *-PRO-***	**17 *-MI-***	**21,38 *-SE***
8	2,75	17,5	26,39

The solutions are in the Table 2:
- The best, regarding the whole cost (solution 2).
- The best, regarding the entire time of all tasks execution (solution 1).
- The best, regarding the entire power consumption (solution The compromise, balancing the values of optimizing criteria (solution 7).

if will be accepted Equation 3:

$$Compromise = minimum \left(\frac{minCost \bullet Cost}{maxCost} + \frac{minTime \bullet Time}{maxTime} + \frac{minPower \bullet Power}{maxPower} \right) \quad (3)$$

3.1 Minimum of cost and Minimum of time

Each of the tables – Table 3 and Table 4 – shows subsequently the best solutions (in Pareto set) regarding cost, time and power consumption. Additionally there is a table which comprises the solutions of balanced costs (compromising solutions) [16].

Table 3. Example for parallel synthesis of Minimum cost and Minimum time (NoT – Number of Tasks)

NoT	Parallel synthesis Minimum of cost			Parallel synthesis Minimum of time		
	Cost	Time	Power	Cost	Time	Power
5	0.5	17	6.47	1.75	4.25	9.56
10	0.75	15.5	15.6	3	3.6	35.47
15	1.5	8.4	54.42	2.75	4.2	77.69
20	1	19	42.64	1.75	12.33	37.23
25	2	15.75	48.51	2	12.25	52.24
30	2.25	18.4	70.51	2.25	14.9	92.18
35	1.5	20.8	114.05	2.75	10.4	173.83
40	2.75	17.75	104.68	2.75	12.6	203.57
45	2.25	24.67	102.02	2.75	14.8	230.11
50	2.25	24.25	108.48	2.75	16.3	242.29
55	2.5	25	164.58	2.75	18	268.59

3.2 Minimum of power consumption and Compromising Solution

The above presented tables – Tab. 3, Tab. 4 show multi-criteria optimization for parallel synthesis of computer systems. As a result of algorithm performance the designer receives a set of optimal solutions in sense of Pareto. The designer has to decide which resources and schedules best fulfills the requirements of the solution. Depending on the system requirements, it is possible to rely on one of the obtained results.

Table 4. Example for parallel synthesis of Minimum power consumption and Compromising solutions (NoT – Number of Tasks)

NoT	Parallel synthesis Minimum of power			Parallel synthesis Compromising solutions		
	Cost	Time	Power	Cost	Time	Power
5	1.75	4.25	9.56	1.75	6.75	9.26
10	3	3.6	35.47	1.5	6.2	35.47
15	2.75	4.2	77.69	2.95	15.5	24.01
20	1.75	12.33	37.21	1.75	12.83	35.45
25	2	12.25	52.24	2	14.5	51.25
30	2.25	14.9	92.18	2.75	16.9	63.58
35	2.75	10.4	173.83	2	18	78.3
40	2.75	12.6	203.57	2.75	17.75	104.68
45	2.75	14.8	230.11	2.25	21.75	99.5
50	2.75	16.3	242.29	2.25	23.88	113.26
55	2.75	18	268.59	2.5	25	164.9

To learn the specification of the solution space for the given problem instance, it is important to provide a sufficiently long list generated of the best solutions [17], 19].
Charts 5, 6 and 7 show the shapes of curves dependent on the number of tasks, presenting compromise solutions against the background of curves dependent on tasks, presenting optimal solutions for minimizing cost, time (speed) and power consumption, respectively.

4. Conclusions

In order to eliminate solution convergence in genetic algorithms [18], [19], we use data structures which ensure locality preservation of features occurring in chromosomes and represented by a value vector. Locality is interpreted as the inverse of the distance between vectors in an n-dimension hyper-sphere.

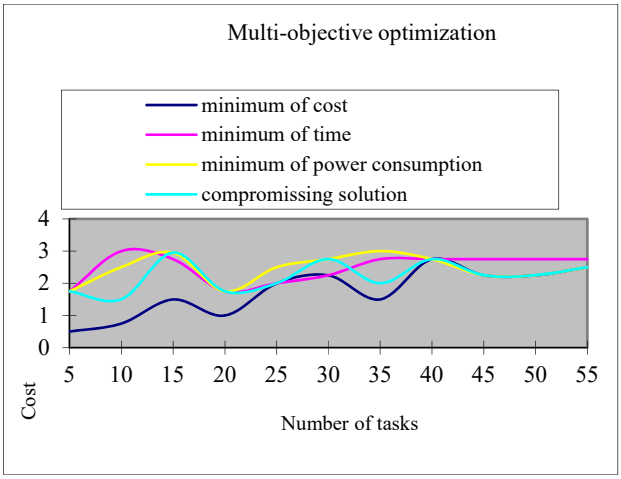

Fig. 5. Chart of compare cost in relation with number of tasks

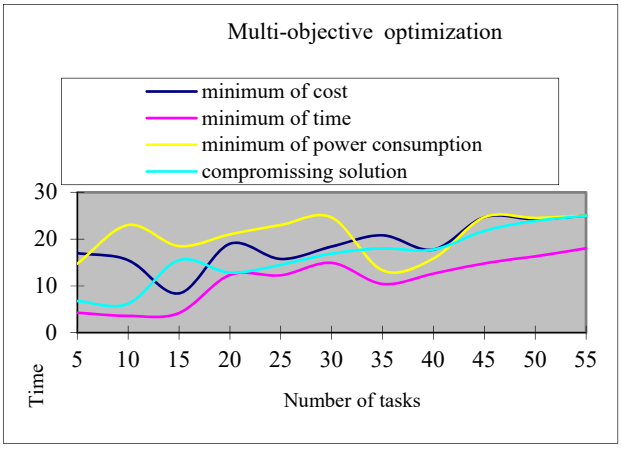

Fig. 6. Chart of compare time in relation with number of tasks

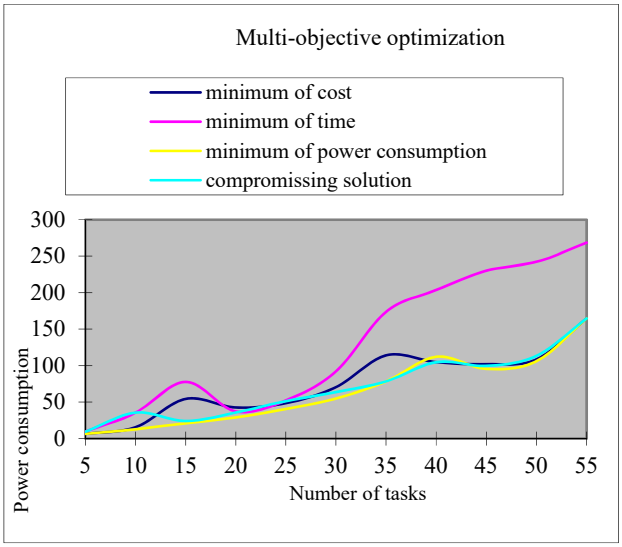

Fig. 7. Chart of compare power consumption in relation with number of tasks

Then, crossing and mutation operators are data exchange operations not between one-dimensional vectors but between fragments of hyper-spheres. Thanks to such an approach, small changes in a chromosome correspond to small changes in the solutions defined by the chromosome. The presented solution features two hyper-spheres: task hypersphere and resource hypersphere. The solutions sharing the same allocations form the clusters together. The introduction of solution clusters separates solutions with different allocations from one another. Such solutions evolve separately, which protects the crossing operation from generating defective solutions. There are no situations in which a task is being allocated to a non-allocated resource. Solution clusters define the structures of the system under construction (in the form of resources for task allocation). Solutions are the mapping of tasks allocated to resources and task scheduling. During evolution, two types of genetic operations (crossing and mutation) take place on two different levels (clusters and solutions). A population is created whose parameters are: the number of clusters, the number of solutions in the clusters, the task graph and resource library. For the synthesis purposes, the following criteria and values are defined: optimization criteria and algorithm iteration annealing criterion if the solution improvement has not taken place, maximum number of generations of evolving within clusters solutions, as well as the limitations – possibly the biggest number of resources, their overall cost, total time for the realization of all tasks, power consumption of the designed system and, optionally, the size of the list of the best and non-dominated individuals. In genetic algorithm with simulated annealing it is also important to define slow cooling of the algorithm (parameters "temperature step" and "cooling coefficient" – depending on the numbers of tasks in the system). Thanks to this, we prevent the population from too big convergence. Algorithm for the lower temperature searches a bigger area in the space of solutions. It has also been noticed that a bigger probability of mutation helps to look for a better system structure, whereas a bigger probability of crossing improves optimization for time criterion.

The problems non-dominated solutions and Pareto optimization of solution spaces for other meta-heuristic algorithms will be studied.

References

[1] Drabowski, M.: Modification of concurrent design of hardware and software for embedded systems – a synergistic approach. In: Grzech, A., Świątek, J., Wilimowska, Z., Borzemski, L. (eds.), Information Systems Architecture and Technology: proceedings of 37th International Conference on Information Systems Architecture and Technology – ISAT 2016, vol. 522, pp. 3-13, Springer, Heidelberg, (2017).

[2] Drabowski, M., Kiełkowicz, K.: A hybrid genetic algorithm for hardware–software synthesis of heterogeneous parallel embedded systems. In: Świątek, J., Borzemski, L., Wilimowska, Z. (eds.), Information Systems Architecture and Technology: proceedings of 38th International Conference on Information Systems Architecture and Technology – ISAT 2017, vol. 656, pp. 331-343, Springer, Heidelberg, (2018).

[3] Błażewicz J., Ecker K., Pesch E., Schmidt G., Sterna M., Weglarz J., Handbook on Scheduling. From A Theory to Practice, Springer Verlag, Berlin, New York, 2019.

[4] Błażewicz J., Ecker K., Plateau B., Trystram D., Handbook on Parallel and Distributed Processing, Springer-Verlag Berlin, Heidelberg, (2000).

[5] Błażewicz J., Drabowski M., Węglarz J.: Scheduling independent 2-processor tasks to minimize schedule length, Inform. Process. Lett. 18, 267-273, 1984.

[6] Błażewicz J., Drabowski M., Węglarz J.: Scheduling multiprocessor tasks to minimize schedule length, IEEE Transactions on Computers, 35(5), 389-393, (1986).

[7] Lee C.Y.: Machine scheduling with availably constraints. In Leung J.Y.T. Handbook of Scheduling, CRC Press, 22.1-22.13, (2004).

[8] Elburi A., Azizi N., Zolfaghri S., A comparative study of a new heuristic based on adaptive memory programming and simulated annealing: The case of job shop *scheduling*, European J. Oper. Res. 177, 1894-1910, (2007).

[9] Saha D., Mitra R.S., Basu A.: Hardware Software Partitioning using Genetic Algorithm, Proc. of the Int. Conference on VLSI Design, 155-160, (1997).

[10] Dick R. P., Jha N. K.: MOGAC: A Multiobjective Genetic Algorithm for the Cosynthesis of Hardware-Software Embedded Systems, Proc. of the Int. Conference on Computer Aided Design, 522-529, (1997).

[11] Dick R. P., Jha N. K.: MOGAC: A Multiobjective Genetic Algorithm for Hardware-Software Cosynthesis of Hierarchical Heterogeneous Distributed Embedded Systems, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 17, no. 10, 920 – 935, (1998).

[12] Ziegenbein D., Richter K., Ernst R., Thiele L.: Teich J., *SPI* – A System Model for Heterogeneously Specified

Embedded Systems, IEEE Trans. on VLSI Systems, Vol. 10, No. 4, 379-389, (2002).
[13] Yhang Z., Dick R. Chakrabarty A.: Energy-aware deterministic fault tolerance in distributed real-time embedded systems, 41st Proc. Design Automation Conf., Anaheim, California, 550-555, (2004).
[14] Schmitz M.T., Al.-Hashimi B.M., Eles P.: Energy-Efficient Mapping and Scheduling for DVS Enabled Distributed Embedded Systems, Proc. of the Design Automation and Test in Europe Conference, 514-521, (2002).
[15] Pricopi, M., Mitra, T.: Task scheduling on adaptive multi-core. IEEE Transactions on Computers C-59, pp. 167-173, (2014).
[16] Agraval, T.K., Sahu, A., Ghose, M., Sharma, R.: Scheduling chained multiprocessor tasks onto large multiprocessor system. Computing, 99 (10), pp. 1007-1028, (2017).
[17] http://www.kasahara.elec.waseda.ac.jp/schedule/.
[18] Montgomery J., Fayad C., Petrovic S., Solution representation for job shop scheduling problems in ant colony optimization, LNCS 4150, 484-491, (2006).
[19] Drabowski, M.: Boltzmann Tournaments in Evolutionary Algorithm for CAD of Complex Systems with Higher Degree of Dependability, In: Zamojski Wojciech, Mazurkiewicz Jacek, Sugier Jarosław, Walkowiak Tomasz, Kacprzyk Janusz (eds.), Advances in Intelligent Systems and Computing, Theory and Engineering of Complex Systems and Dependability: Proceedings of the Tenth International Conference on Dependability and Complex Systems DepCos-RELCOMEX- 2015, vol. 365, pp. 141-152, Springer, Heidelberg, (2015).

Mieczyslaw Drabowski, Professor at Cracow University of Technology (CUT), works at Faculty of Electrical and Computer Engineering. He received the M. Sc. degree in electrical engineering, specialty: automatic control and communication, from AGH University of Science and Technology, graduated mathematic from Jagiellonian University, received the Ph. D. degree (with distinction) in computing science from Poznan University of Technology and Sc. D. degree in computing science from West Pomeranian University of Technology. He has eighteen years of industrial experience and implementations in software engineering, testing of computer electronic circuits, in micro diagnostics for computer systems, in designing of disks storages and their controllers and in designing personal computers and their network. He is the member of several editorial boards, among others Scientific Journals International, Journal Software: Practice and Experience, International Journal of Electronics and Telecommunications, Technical Transactions (Computer Science and Information Systems), International Conference on Dependability and Complex Systems. His research interests include operating systems, software engineering, task scheduling in computer systems, synthesis of computer systems, allocation for tasks and resources in parallel and distributed multiprocessors systems, dependable and fault tolerant systems, computational intelligence and computer system performance evaluation. He is author and co-author of 3 monographs and over 100 papers in major professional journals and conference. He lectures at the Cracow University of Technology, teaching engineering, master's and doctoral studies, supervising diploma theses. Lately, he was the deputy dean and head of Department of Theoretical Electrical Engineering and Computing Science in Faculty of Electrical and Computer Engineering, Cracow University of Technology.