Hyperspheres and Clusters in Hybrid Algorithm for Aided Design of Systems

Mieczyslaw Drabowski[*] and Edward Wantuch[**],

[*]Cracow University of Technology, [**]AGH University of Science and Technology

Summary
This paper presents a new approach for modeling and aided design of heterogeneous system. This approach based on hybrid algorithm: genetic method and simulated annealing strategy with Boltzmann tournament selection. In order to eliminate solution convergence we use following data structure: n-dimension hyper-spheres and clusters for separates solutions with different allocations functions to resources.

Key words:
design of systems, genetic method, hyper-spheres, clusters, Boltzmann tournament

1. Introduction. Hybrid method for aided design of systems

The aim of high-level design of heterogeneous systems is to find an optimal solution which will meet the requirements and constraints enforced by the given specification of the system. The following criteria of optimality are usually considered: the cost of system implementation, its operating speed, power consumption and dependability. The specification describing a computer system may be considered then as a set of interactive functions.

The function partition of a so defined system into hardware and software is the basic problem of synthesis. Such a partition is significant, because every computer system must be realized as result of hardware implementation of its certain tasks. The constructed system consists of peripheral elements and software components, selected from the available pool of hardware modules. The system is specified as a set of requirements and constraints to be met. In general, each requirement may be realised by hardware elements or software components executed by universal processors and memory. Obviously, at this stage of design, one must look into appropriate system constraints and criteria of optimal system operation. Accordingly, the key issue in the synthesis is an efficient partitioning of system resources due to their hardware and software implementation, providing the fulfilment of all requirements and the minimum implementation cost.

Such partitioning methodology may accept, as a starting point, the assignment of hardware implementation to all system functions and for further optimization of project costs may search for possibilities of replacing certain tasks realized by hardware with their software equivalents.

Other methods of the resource partitioning may start with an exclusive software implementation and further search for implementation of certain tasks by hardware. In both approaches the objective is to optimize the implementation cost of the designed system for the same tasks, i.e. minimization of peripheral solutions especially specialized ones. Obviously, requirements and constraints, particularly those regarding time and dependability have a decisive influence upon selection of necessary hardware components.

The measure for an efficient implementation of a computer system is the degree of its modules utilization, minimized idle-time of its elements and maximized parallel operation of its elements.

A non-optimum system contains redundant modules or modules that are excessively efficient in comparison with the needs defined by the tasks which, consequently, increases the system cost. In high-level synthesis, the optimization of the designed system costs, speed and power consumption is usually an iterative process, requiring both changes in the architecture and task scheduling. Therefore, an optimum system should be created as a compromise between the projects: system control and its hardware organization.

System synthesis is a multi-criteria optimization problem. As for the optimality criteria for the system to be designed, we shall assume its minimum cost, maximum operating speed minimum power consumption and maximum dependability.

We will apply multi-criteria optimization in sense of Pareto [1]. 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 solution 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

Manuscript received October 5, 2013
Manuscript revised October 20, 2013

set of solutions; nevertheless, in case of a set of suboptimal solutions, they constitute one form of higher order optimal set in sense of Pareto and they give, by the way, access to the problem shape of Pareto optimal set of solutions.

Due to the fact that synthesis problems and their optimizations are strongly NP-complete [2] we suggest meta-heuristic approaches, in this paper, hybrid: genetic with simulated annealing [3].

From this section, input the body of your manuscript according to the constitution that you had. For detailed information for authors, please refer to [1].

2. Clusters and hyper-spheres in hybrid algorithm

In order to eliminate solution convergence in genetic algorithms, 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 [4]. 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 solution defined by the chromosome. The presented solution features two hyper-spheres: task hyper-sphere and resource hyper-sphere. The solutions sharing the same allocations form the so-called clusters. 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.

Application of the same resources clusters aims at separating solutions [5] of different allocation from one another. Such solutions evolve separately which protects crossbreeding operation from producing defective solutions; the situation where the task will be attached to the recourse which has not been allocated does not happen. Figure – Fig. 1 – illustrates what crossing of two solutions: *1* and *2* would be if solutions of different allocations to resource were not separated. In solution *1* – tasks *A* and *B* are conducted on processor *P1*; in solution *2* tasks *A* and *B* are conducted on processor *P2*. After the crossing new solutions: *3* and *4* are created. As a result of this operation, task *A* in solution *3* has been allocated to processor *P2* which is not used in this solution. In solution *4* there is a similar situation. Then as a result of crossbreeding operation incorrect solutions would be created.

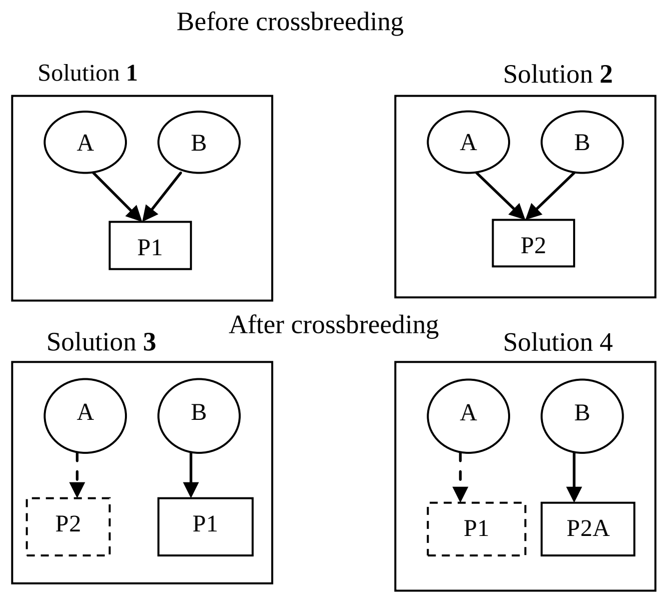


Fig. 1. Faulty crossbreeding of two solutions

Hyper-sphere of system features contains the information related to the similarities between features of resources.

This structure contains the following information:

- Each gene in chromosome is described by feature vector (e.g. hardware resource – by the cost, power consumption, speed and degree of reliability). Hyper-sphere stores the distance of feature vectors from the centre of hyper-sphere. The data is needed while crossbreeding in order to define which information will be transformed into successors.
- Co-ordinate axes of the centre of hyper-sphere – assigned by the set of feature vectors – are used at constructing hyper-plane dividing hyper-sphere into two parts.
- Length of the diameter – the distance between the most distant feature vectors –used at constructing hyper-plane.

- The coefficients are calculated during each crossbreeding. After the operation of crossbreeding, two new individuals (successors) are created. Each of the successors comprises one part of the first parent‘s chromosome and the second part of the other.

System structure is represented by the set of data tables. Each chromosome consists of a vector describing recourse allocation and vector describing task allocation and scheduling. Each gene comprises multidimensional information. Resource processor is described by cost, speed and power consumption, reliability. During crossbreeding it comes to the exchange of data between the tables.

Skilful performance of evolutionary algorithm needs storing the locality in data structure representing solution. The exchange of data between individuals (crossbreeding) should separate the information describing similar features of system structure more rarely than information describing entirely different features. Small changes in chromosome should reflect small changes in the solution which is represented by the chromosome.

Introducing linear order into multidimensional information destroys locality. To solve this problem we used data structures representing hyper-sphere. Multidimensional information is recorded in the shape of vector. Locality is interpreted as the inverse length of n-dimensional vectors inside n-dimensional hyper-sphere.

The most distant vectors mark diameters and the centre of hyper-sphere. The example of two-dimension hyper-sphere is illustrated by Fig. 2.

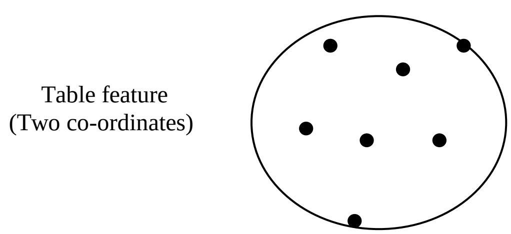


Fig. 2. Two dimensional hyper-sphere (circle) – resources are described by two features here, (e.g. speed and cost)

The presented algorithm stores hyper-sphere of resources is four-dimensional hyper-sphere representing the dependability of resource features. Each of the resources may be defined by the following coordinates: reliability, cost, speed and power consumption.

2. Resources partition algorithm for aided design of systems

It tasks into account the data of task graph, pool of resources as well as criteria of optimality. The aim of this algorithm is to determine resources for execution all tasks in accordance with the criteria of optimality.

Initialization of algorithm

The aim of initializations of algorithm is to construct such structure of system which is the initial, the simplest and based on available resources and at the same time satisfying the demands of functions and given criteria.

Algorithm executes the following steps:

- Construction of tasks graph.
- Creating resources.
- Creating population.
- Initialization of hyper-sphere.
- Population initialization.
- Solution evaluation.
- Cluster evaluation.

Construction of task graph

On the basis of input data (available from system specification of type complex operations) the structure describing the digraph of task graph is created. The digraph of tasks represents the functionality of the system. After graph creation, nodes are sorted. Topological order defines the position of tasks in graph. Nodes located close to one another are situated similarly close on the list of typologically sorted nods. This information is applied by algorithm of task scheduling in accordance with graph levels.

The first step of algorithm is graph searching by method of DFS (Depth First Search) in order to indicate the tasks levels in graph. The nod level is perceived as the longest path from the nod without predecessors to the examined nod. This information is used in algorithm of task scheduling. If tasks are scheduled according to the levels in graph (in an increasing order), keeping constraints' sequence is preserved. The next step is to search the graph by algorithm BFS (Breadth-First Search). The nods are given indexes calculated during the algorithm run. These indexes store information which applies to task parallelism in the graph. This information is used in algorithm as one of the task dependence features in the graph.

Creating resources

On the grounds of data input (specification of resource pool), the set representing the accessible resources for system synthesis is created. This set makes the information about recourses (such as cost of processors, cost of operating memory, executing times, speed, power consumption, sum of power consumption of all the tasks, degree of reliability) available.

Creating population

Parameters of population are:

- Number of clusters in population.
- Number of solutions in clusters.
- Digraph of tasks – the functionality of system.
- Accessible resources.
- Criterion of halt – defines the numbers of algorithm loops when the solution improvement has not happened.
- Map of criteria – defines, which of the criteria of optimization will be considered while searching for the optimum solution.
- Considering the costs of operating memory during cost optimization.
- Maximum number of evolving solution generations inside the clusters.
- Constraints – maximum number of processors, maximum cost, maximum time, maximum power consumption.
- Size of the list of the best individuals. The best non-dominated individuals are enrolled onto the list. The longer the list is, the more individuals can be remembered by algorithm. The list is the of FIFO queue.

Clusters and solutions in cluster are created as well as hyper-spheres of resources and task graph. Also, this stage the parameter “global temperature” is introduced.

Initialization of hyper-sphere

Two hyper-spheres are created: resources and task graph. The hyper-sphere of processors is of 1 to 4 dimensions. The dimension depends on the number of optimized features (cost, time, power consumption, reliability). Hyper-sphere for the graph of tasks is two dimensional.

Definitions of hyper sphere consist of:

- Filling multi-dimensional vectors with data defining a given object (resources, tasks).
- Calculating the diameters of the hyper spheres, i.e. the distance between the two most remote points and determining the hyper sphere center on the basis of the extreme coordinate.
- Defining the centre of hyper-sphere on the grounds of coordinate axes.

Population initialization

Clusters and solutions are initialized at random.

Algorithm of allocation initialization

- For each task, a resource capable of completing the task is selected.
- If the resource is allocated, the algorithm proceeds to the next task.
- A resource capable of completing the task is selected and allocated.

Initialization algorithm of allocating tasks to resources

- We assign resources for each task which are allocated and capable of task completing.
- Resource type and number are randomly assigned to the tasks.
- Task scheduling by ASAP (As Soon As Possible) algorithm is initialized:
    - We examine subsequently the tasks on the graph levels; such sequence ensures the intact of sequence constraints.
    - We allocate it to the list of scheduling according to the previously allocated processor.

Solution evaluation

The following values are calculated depending on optimization criteria:

- Resource cost (processors: general, dedicated and operating memory).
- Task completion time.
- Power consumption.

The whole cost is the cost sums of allocated recourses; the entire time of executed tasks is the time of completing all the tasks on all the allocated recourses, the whole power consumption is the sum of power consumption taken by the selected recourses. If, for the representing individual, the unlimited solution of the optimization criteria excesses the maximal available number of value, the individual is fined. The suitable value for this individual is multiplied and the chance of survival definitely decreases. As a result of the above mentioned operations, we receive the table comprising the value of optimized criteria (time, cost, power consumption). The following stage is to determine solution ranking. Ranking of a solution is the number of solutions in population which do not dominate this solution. Solution is dominated if each of its optimized criteria value is lower or equal to the value for the dominated solution (optimization in sense of Pareto).

Cluster evaluation

Solution cluster ranking is created. Cluster ranking is the sum of the solution ratings within the cluster.

Main algorithm

The input data for resource partition (after initialization) are the task digraph, the library of available resources and the optimization criteria, and its goal is to divide tasks into the software and the hardware part and to select resources for the realization of all the tasks with established optimization criteria. The diagram of the algorithm of resource partition is showed on Fig. 3.

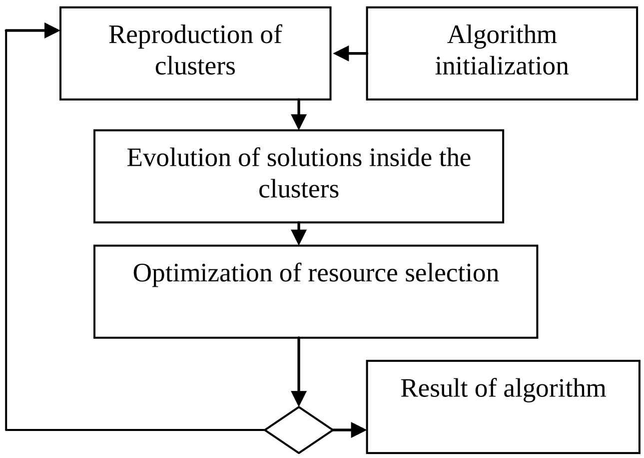


Fig. 3. Algorithm of resource partition

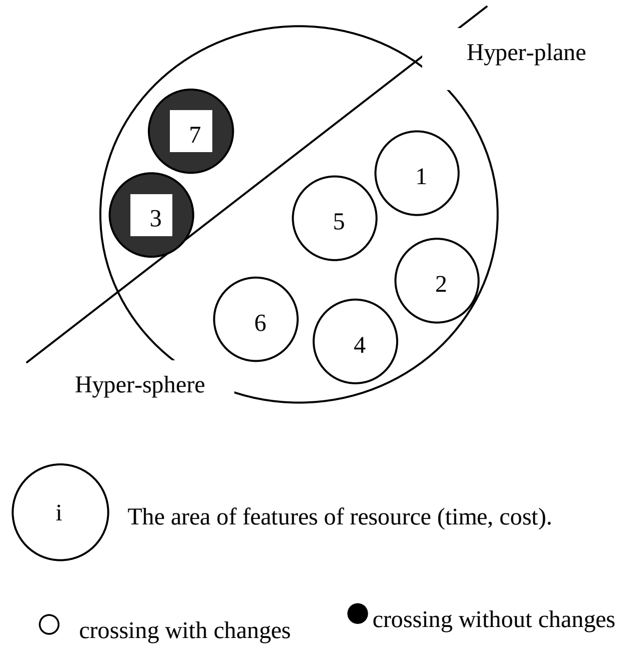
Cluster reproduction

Clusters are reproduced with the use of genetic operators: crossing and mutation. At the reproduction stage, the cluster population is doubled and its initial size is restored at the elimination stage. This method was introduced arbitrarily and ensures that within a population some new individuals appear and fight for survival with their parents. The mutation operator creates one, and the crossing operator two new clusters. The likelihood of using either of the genetic operators is defined by the algorithm parameters.

Genetic operators:

- **Cluster mutation operator** is to mutate allocation tables in the following way: a cluster with identical likelihood is picked at random and copied. The number of the resource which will be mutated in a new cluster is picked randomly. Then, a number in the 0-1 range is picked – if the number is smaller than the parameter of global temperature, the resource is added; otherwise it is subtracted. Adding resources is limited by the parameter of constraints: *maximum resource number parameter*. At the beginning of the algorithm operation, resources will be added to the structure of the system. As the algorithm approaches the end of the run defined by the cooling process, resources will be subtracted. This method aims at creating a cost-economical structure.
- **Cluster crossing operator** is to pick randomly two clusters and copy them. Crossing is achieved through cutting the resource hyper-sphere with a hyper-plane. The information contained on "one side" of the hyper-plane is exchanged between clusters – Fig. 4.

The example (Example *) shows the result of the performance of crossing tables for resource selection with the use of resource hyper-sphere. Each resource is described with help of two features (e.g. cost and speed).



4. The crossing operator with the hyper-sphere

Example *:

Table 1 of allocation of resources before crossing

PE 1	PE 2	PE 3	PE 4	PE 5	PE 6	PE 7
1	2	**0**	1	0	2	**5**

Table 2 of allocation of resources before crossing

PE 1	PE 2	PE 3	PE 4	PE 5	PE 6	PE 7
3	0	**2**	2	2	1	**0**

Table 1 of allocation of resources after crossing

PE 1	PE 2	PE 3	PE 4	PE 5	PE 6	PE 7
1	2	**2**	1	0	2	**0**

Table 2 of allocation of resources after crossing

PE 1	PE 2	PE 3	PE 4	PE 5	PE 6	PE 7
3	0	**0**	2	2	1	**5**

The similarity of resources is interpreted as the distance between the tables describing resource features in two-dimensional space. During the crossing, two-dimensional hyper-sphere is cut (circle) by hyper-plane (straight line). As a result of this operation, the information contained on one side of the line is exchanged between the tables of resource selection.

The algorithm for cutting the hyper-sphere with a hyper-plane

- Determining the cutting of hyper-plane by picking ***n*** points inside an n-dimensional hyper sphere.
  - Creating a random permutation of n dimensions, e.g. for n = 3, the permutation can be (2, 1, 3).
  - Constructing the point displacement vector in respect to the hyper- sphere center for ***n*** dimension:
    - Square coordinates are picked according to dimension permutations, e.g. for three dimensions with the permutation (2, 1, 3):
      y^2 = rand () % r^2, x^2 = rand () % ($r^2 – y^2$), z^2 = rand () % ($r^2 – (y^2 + x^2)$)
      Where:
      r – hyper-sphere radius, (x, y, z) are the coordinates of the constructed point in a three-dimensional space.
    - The roots of square coordinates are calculated.
    - A coordinate radical sign is picked.
    - The hyper-sphere center coordinates are added to the new point resulting in obtaining a new point inside the n-dimensional hyper-sphere.
- The equation of the hyper-plane cutting the hyper-sphere is calculated and the obtained system of linear equations is solved.

After creating new clusters, initiating algorithm of task allocation and initiating algorithm of scheduling are put into motion.

Example **

Vectors of resources: P_1 (5, 2), P_2 (7, 6), P_3 (11, 4), P_4 (9, 3), P_5 (3, 3) – e.g. by the speed and cost.
Construction on hyper-sphere:
length of diameter – the distance between the most distant feature vectors - P_3 i P_5 - $d^2 = (11 – 3)^2 + (3 – 4)^2 = 65$ and hyper-sphere center S (½*(11 + 3), ½ *(4 + 3)) = (7, 3.5), r = ½ * d, r^2=16.25
x'^2 = rand() * r^2, y'^2 = rand() * ($r^2 – x^2$) $x'^2 = 16$, $y'^2 = 0.16$ and $x''^2 = 9$, $y''^2 = 7.25$
x' = 4, y' = -3 and x" = -3, y" = 2.69, new vectors: P' (11, 3.9) i P" (4, 6.19)
and hyper-plane: y = a * x + b with a = - 0.314 i b = 4.84
The example shows the result of the performance of crossing tables for resource selection with the use of resource hyper-sphere. Each resource is described with help of two features (e.g. cost and speed). The similarity of resources is interpreted as the distance between the tables describing resource features in two-dimensional space. During the crossing, two-dimensional hyper-sphere is cut (circle) by hyper-plane (straight line). As a result of this operation, the information contained on one side of the line is exchanged between the tables of resource selection - P_1 i P_5.

Evaluation of solutions

This is the same algorithm which is employed in the initialization of resource partition algorithm.

Saving the best solutions

After the solution reproduction, a new procedure is called to save globally non-dominated solutions generated during evolution.
This procedure executes:

- Searches for non-dominated solutions in the present generation.
- Creates the ranking of the best solutions saved so far and in the present generation.
- Saves the non-dominated solutions from both the "old" and the "new" solutions.
- Deletes the solutions saved in the past if they are dominated by new solutions; if there is more than one solution that's all optimized criteria values are identical, only one of those solutions is saved (the "newest" one).
- If the new solutions dominate none of the ones saved in the past, the population is not improved.
- Algorithm can remember the defined by algorithm parameter numbers of non-dominated solutions.

Evaluation of clusters

This is the same algorithm which is used during initialization of resource partition.

Cluster elimination

At this stage of the algorithm, half individuals are removed from the population. The initial number of individuals is restored. The elimination of individuals is carried out with use of Boltzmann tournament selection strategy.

Boltzmann tournament

The winner of the tournament is chosen on the base of calculating the following equation:

$$f(x) = (1 + e^{\frac{(r1-r2)}{T}})$$

Where:

- r1 – Ranking of the first solution.
- r2 – Ranking of the second solution.
- T – Global temperature.

The values of this function are numbers of the range ***< 0, 1 >***. In order to assign the winner of the tournament, we choose the number of the range ***(0, 1)***. If it is larger than the one enumerated out of the formula, the individual of ranking ***r1*** *is the winner*. Otherwise, the second individual (of ranking ***r2***) is the winner.

The analysis of the results of tournament can be carried out on the basis of function graph Fig. 5.

$$f(x) = (1 + e^{x})^{-1}$$

Where:

$$x = \frac{(r1 - r2)}{T}$$

If ***r1*** < ***r2***, then *x* is negative, in this case when the temperature is high, there is a larger probability that an individual of rank ***r1*** *will* win the tournament than when the temperature is lower. For low temperatures an individual of rank ***r2*** most frequently will be the winner.

If ***r1*** > ***r2***, then ***x*** is positive, in case when the temperature is high, there is a larger probability that an individual ***r2*** *will* win the tournament than for lower temperatures. For low temperatures, an individual of rank ***r1*** most frequently will be the winner.

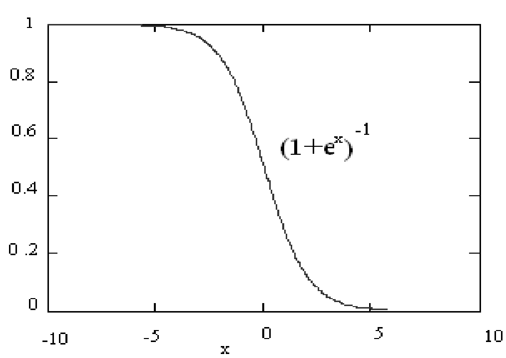


Fig.5. The chart of probability of winning Boltzmann tournament depending on global temperature

Report of algorithm

If the number of generations during which the improvement of individuals did not happen exceeds the number given in the criterion of halt, algorithm finishes its working. Non- dominated individuals in the whole scale of evolution are recorded in a report.

3. Conclusions

Genetic algorithm with hyper- spheres and clusters obtained solutions with lower cost than algorithm without this of structures for all instances of problems of design of systems. Solutions obtained with this algorithm feature shorter schedule length when compare to solutions of algorithm without hyper-spheres and clusters.

Selected results: schedule length minimization (speed of system) and cost minimization - Table 1 and Chart 1 and Chart 2.

Applications of clusters and hyper-spheres significantly improved results obtained.

Table 1. Schedule length (speed of system) and cost of system for algorithms of synthesis: without hyper- spheres and with hyper-spheres

Number of tasks	without hyper-spheres		with hyper-spheres	
	Schedule length	Cost	Schedule length	Cost
10	1,33	10,55	0,80	9,10
20	1,90	13,35	1,40	3,25
30	2,50	13,30	2,00	4,50
40	2,30	13,40	2,50	3,25
50	2,90	22,00	3,10	3,75
75	4,60	21,65	4,40	4,75
100	12,40	19,55	5,70	5,00

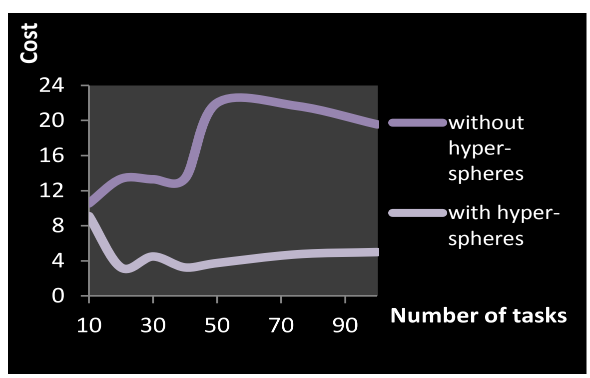


Chart. 1 Algorithms without hyper-spheres and with hyper-spheres in synthesis of system for criterion of cost

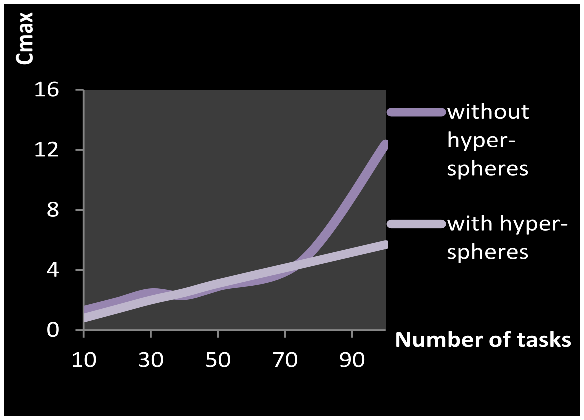


Chart. 2 Algorithms without hyper-spheres and with hyper-spheres in synthesis of system for criterion of speed (schedule length C_{max}

References

[1] Drabowski M., Wantuch E., *Artificial Intelligence in synthesis of heterogonous of computer system,* International Symposium on Application of System Theory, Zakopane, 2011.

[2] Garey M. R., Johnson D. S., *Computers and intractability, A guide to the theory of NP-complete,* San Francisco, 1979.

[3] Drabowski M., *Parallel synthesis of computer systems,* Monographs 225, AGH-UST, Krakow, 2011.

[4] Dick R. P., Jha N. K., *COWLS: Hardware – Software co-synthesis of distributed wireless low power client-server systems,* IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems, Vol. 23, No. 1, 2-16, 2004.

[5] Dick, R.P., Laskhminarayana G., Raghunathan A., Jha N. K., *Analysis of Power |Dissipation in real time operating systems,* IEEE Transaction on Computer Aided Design of integrated circuits and systems, Vol. 22, No. 10, 615-627, 2003.

[6] Drabowski M., Wantuch E., *Scheduling in Manufacturing Systems Ant Colony Approach*. [in:] Ant Colony Optimization Techniques and Applications, Intech Croatia, Austria, India, 2013.

Mieczyslaw Drabowski, Assistant Professor and Head of Laboratory of Computing Sciences, Faculty of Electrical and Computer Engineering, Cracow University of Technology, received the M. Sc. degree in automatic control and communication from AGH University of Science and Technology, graduated mathematic from Jagiellonian University in Krakow and received the Ph. D. degree (with honors) in computing science from Poznan University of Technology, in 1977, 1979 and 1986, respectively.

Currently he is member of several editorial boards, among others Scientific Journals International, International Association for Development of the Information Society (IADIS), and International Association of Science and Technology for Development (IASTED) on Artificial Intelligence and Soft Computing. His research interests include schedule, assignment and allocation for tasks and resources, dependable and fault tolerant systems, artificial intelligence, operating systems and software engineering, author and co-author of 3 monographs and over 60 papers in major professional journals and conference proceedings. Dr. Drabowski is a member of the council of the Polish Information Processing Society.