Genetic Algorithm of Resource Partition and Task Scheduling

Mieczyslaw Drabowski

Cracow University of Technology
Warszawska 24 Krakow 31-155 Poland
drabowski@pk.edu.pl

***Abstract*:** The paper presents an innovative approach to solving the problems of computer system synthesis based on genetic methods assisted with simulated annealing strategy. We describe algorithm realizations aimed to optimize resource partition and task scheduling, as well as the adaptation of those algorithms for coherent synthesis realization. We then present selected analytical experiments proving the correctness of the coherent synthesis concept and indicate its practical motivations.

***Keywords*:** task, resource, allocation, genetic, coherent, synthesis.

1. Introduction

The goal of high-level synthesis of computer systems (i.e. systems of type the complex of resources and operations) is to find an optimum solution satisfying the requirements and constraints enforced by the given specification of the system. The following criteria of optimality are usually considered: costs of system implementation, its operating speed, power consumption and dependability. A specification describing a computer system may be provided as a set of interactive tasks (processes, functions).

The partition of the functions between hardware and software is the basic problem of synthesis. Such partition is significant, because every computer system must be realized as result of hardware implementation for its certain tasks.

In the synthesis methods so far, the software and hardware parts were developed separately and then connected in process the co-called co-synthesis, which increased the costs and decreased the quality and reliability of the final product.

The resources distribution is to specify, what hardware and software are in system and to allocate theirs to specific tasks, before designing execution details.

The problems of tasks scheduling are one of the most significant issues occurring at the procedure synthesis of operating systems responsible for controlling the distribution of tasks and resources in computer systems.

The objective of this research is to present the concept of coherent approach to the problem of system synthesis, i.e. a combined solution to task scheduling and resource partition problems. The model and approach are new and original proposals allowing synergic design of hardware and software for performing operations of the computer system. This is approach, which we called a par-synthesis (coherent co-synthesis).

This research shows the results selected of computational experiments for different instances of system par-synthesis problems proving the correctness of the coherent synthesis concept and shows the methods solving this problems.

Due to the fact that synthesis problems and their optimizations are NP-complete we suggest meta-heuristic approach, i.e. genetic with simulated annealing.

Coherent co-synthesis of computer systems, as well as synergic design methodology their structures and scheduling procedures may have practical application in developing the tools for automatic aided for rapid prototyping of such systems.

2. Coherent synthesis of computer system

2.1 The classical process of computer system synthesis

The classical process co-synthesis [2], [14], [22] – hardware and software – for computer system consists of the following stages (Fig. 1.1):

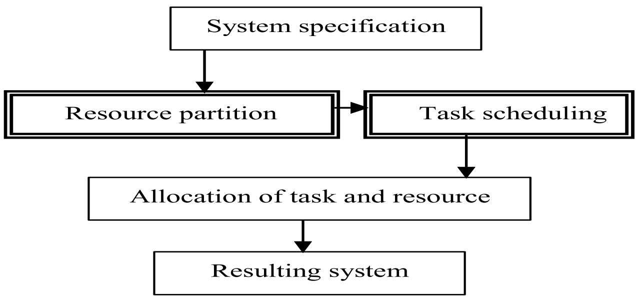


Fig. 1.1. The process co-synthesis

1. Specification of the designed system in terms functional and behavioural – requirements and constraints analysis. The system description in an high-level language, abstracting from the physical implementation.
2. Resource partition – architecture development.
3. Task scheduling – system control development.
4. Allocation the system functions to the architecture elements – generating the system modular architecture, control adaptation and the whole system integration.

The system being constructed consists of hardware elements and software components performed by selected hardware modules. The system is specified by a set of requirements to be met. In general, each requirement may be satisfied by hardware elements or software components executed by universal processors and memories. Obviously, at this stage of design, one must take into account appropriate system constraints and criteria of optimal system operation. Accordingly, the key issue in the synthesis is efficient partitioning of system resources due to their hardware and software implementation, providing fulfilment of all requirements and the minimum implementation cost.

Such partitioning methodology [17] may accept, as a starting point, assignment of the hardware implementation to all system functions and further optimization of project costs, search for possibilities of replacing certain tasks realized by hardware with their software equivalents. Other methods [20] of the resources partitioning start with an exclusive software implementation and further search for implementation of certain tasks by hardware. In both approaches the objective is optimization of the implementation cost of the same tasks, i.e. in particular minimization of the execution time by specialized hardware [3]. Obviously the requirements and constraints, especially those regarding time and power consumption, have 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 [21].

A non-optimum system contains redundant modules or modules that are excessively efficient in comparison to the needs defined by the tasks what, 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 [23]. That is, why an optimum system may be created as a compromise between the system control algorithm and its hardware organization.

2.2 The general model for the problem of system synthesis

System synthesis is a multi-criteria optimization problem. The starting point for constructing our approach to the issues of hardware and software synthesis is the deterministic theory of task scheduling [4], [7], [25]. The theory may serve as a methodological basis for multiprocessor systems synthesis.

Accordingly, decomposition of the general task scheduling model is suggested, adequate to the problems of computer system synthesis. From the control point of view such a model should take into account the tasks, which may be either preemptable or nonpreemptable. 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 a case the schedules are called to be preemptive. Otherwise, tasks are nonpreemptable and schedules nonpreemptive.

Preemptability of tasks in our approach cannot be a feature of the searched schedule – as in the task scheduling model so far. The schedule contains all assigned tasks with individual attributes: preemptive, nonpreemptive. From the point of view of the system synthesis, the implementation of certain tasks from the given set must be nonpreemptible, for the other may be preemptible (what, in turn, influences significantly selection of an appropriate scheduling algorithm) [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 to universal and specialised hardware components. Accordingly, we shall discuss the system of type the complex of resources and operations:

$$\sum = \{ \mathbf{R}, \mathbf{T}, \mathbf{C} \} \qquad (1)$$

where:
R – is the set of resources (hardware and software),
T – is the set of the system's tasks (operations),
C – is the set of optimization criteria for the system's behaviour and structure.

Resources. We assume that processor set $P = \{P_1, P_2, \ldots, P_m\}$ consists of *m* elements and additional resources $A = \{ A_1, A_2, \ldots, A_p\}$ consist of *p* elements.

Tasks. We consider a set of *n* tasks to be processed with a set of resources. The set of tasks consists of *n* elements $T = \{T_1, T_2, \ldots, T_n\}$. A feasible schedule is optimal, if its length is minimum and it is implemented using minimum resource cost.

Each task is defined by a set of parameters: resource requirements, execution time, ready time and deadline, attribute - preemptable or nonpreemptable. The tasks 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 they are a set of independent tasks.

Optimality criteria. As for the optimality criteria for the system being designed, we shall assume its minimum cost, maximum operating speed and minimum power consumption.

The proposed model may be used for defining various synthesis problems for optimum computer systems.

The model of a system in our approach, [9], [11] typical for the theory of task scheduling, consists of a set of requirements (operations, tasks) and existing relationships between them (related to their order, required resources, time, readiness and completion deadlines, preemptability/nonpreemptability, 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 and defining the level of resource co-sharing and the degree of concurrency in task performance.

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 integrated circuits (readily available on the resources pools or designed in accordance to the suggested characteristics) [18]. Software modules of the system are generated with the use of software engineering tools. Appropriate processors shall be taken from the resource pool. Synthesis of a system may also provide a system control, create an interface and provide synchronization and communication between the tasks implemented by software and hardware [11].

The system synthesis, i.e. defining system functions, identifying resources, defining control should be implemented in synergy and be subject to multi-criteria optimization and verification during implementation.

2.3 The coherent process of system synthesis

Modeling the joint search for the optimum task schedule and resource partition of the designed system into hardware and software parts is fully justified. Simultaneous consideration of these problems may be useful in implementing optimum solutions, e.g. the cheapest hardware structures. Synergic approach enables also performing of all assigned tasks with the minimum schedule length. With such approach, the optimum task distribution is possible on the universal and specialized hardware and defining resources with maximum efficiency.

We propose the following schematic diagram of a coherent process of systems synthesis [10], (Fig. 1.2). The suggested coherent synthesis consists of the following steps:

1. specification of requirements for the system to be designed and its interactions with the environment,
2. specification of tasks, including evaluation of task executive parameters using available resources (e.g. execution times),
3. assuming the initial values of resource set and task scheduling – initial resource set and task schedule should be admissible, i.e. should satisfy all requirements in a non-optimum way,
4. task scheduling and resource partitioning,
5. evaluating the operating speed and system cost, multi-criteria optimization,
6. the evaluation should be followed by a modification of the resource set, a new system partitioning into hardware and software parts (step 4).

Iterative calculations are executed till satisfactory design results are obtained – i.e. optimal (or sub-optimal) system structure and schedule. The designed system should be fast and cheap.

3. The genetic method for coherent synthesis of computer system

This chapter presents a coherent approach to solving the problems of computer system synthesis based on genetic method assisted with simulated annealing strategy. We describe algorithm realizations aimed to optimize resource partition and task scheduling, as well as the adaptation of those algorithms for coherent co-synthesis realization. We then present selected analytical experiments proving the correctness of the coherent synthesis concept and indicate its practical motivations. Due to the fact that synthesis problems and their optimizations are NP-complete [6], [15] we suggest meta-heuristic approach, genetic with Boltzmann tournament selection strategy [1], [12], [24].

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 hypersphere. Then, crossing and mutation operators are data exchange operations not between one-dimensional vectors but between fragments of hyperspheres. 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 hyperspheres: task hypersphere and resource hypersphere.

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).

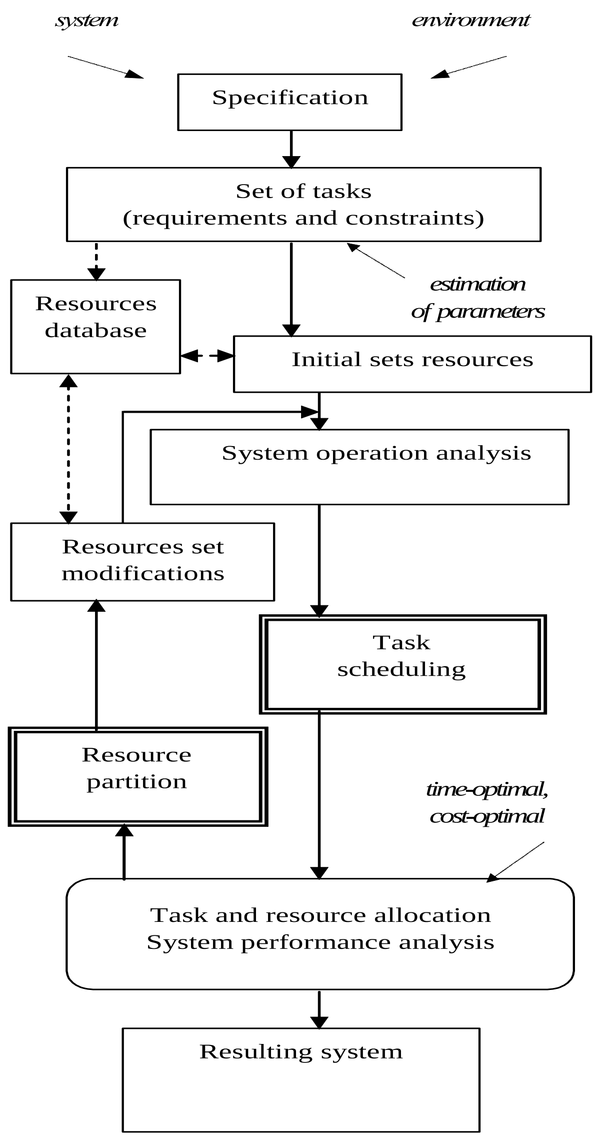


Fig. 1.2 . The coherent process of computer system synthesis

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 solution improvement has not taken place, maximum number of generations of evolving solutions within clusters, as well as the limitations - 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.

3.1 Data structures

3.1.1 *The structure "Population"*

This structure contains information about individuals' population:

- The table of clusters of solutions.
- Hyperspheres the resources and graph of tasks.
- Number of generation of clusters.
- Number of generation of solutions in cluster.
- Maximum quantity of generations of evolving solutions inside the clusters.
- Criterion of stop of evolution - the maximum quantity of generations without improvement of solution.
- Number of generations without obtainment of improvement of solution.
- Keeping table the the best, dominated encountered during evolution solution.
- Map of costs of optimization.
- Criterion of optimization - the maximum quantity of processors, maximum cost, maximum time, power maximum consumption.
- Dimension the list of the best solutions.
- Probability of crossing of individuals. The of mutation even probability is 1 - the probability of crossing.

3.1.2 *The structure "Clusters of solutions"*

The structure contains the information about cluster of solutions possessing the same the allocation of resources:

- Area describing the allocation of resources.
- Total price of allocated processors.
- Ranking of clusters (the sum of rankings of solutions inside the cluster).

Use of clusters of solutions about the same allocation has on aim the separating from me the solutions about different alokacjach. Solutions such evolve separately. In it secures oneself then the operation of crossing before production the defective solutions. It does not come to situation such that task be becomes attached to supply which he does not be allocate.

3.1.3 *The structure " Solutions in demand"*

The structure contains the information about the outcome structure and functionality of architecture of system:

- The area describing the attributing to resources the tasks.
- The table of optimized costs.
- Ranking of solution (the quantity of solutions in population which did not dominate this solution)

3.1.4 *The structure "Allocation of resources"*

The structure contains about allocated resources in frames of cluster of solutions:

- The table of solutions about the same allocation of supplies.

3.1.5 *The structure "Attributing to resources tasks"*

The behaviour of tasks describes in system (attributing, schedule):

- The table of list describing order in a row on individual processors tasks. Every list responds one allocated processor.
- The table including the times of beginning and end of executing the tasks.
- The table the describing allotment of tasks to allocated resource.

3.1.6 *The structure "Graph of tasks"*

The structure contains the information about of graph of tasks describing the functional requirements of system:

- Number of tasks in vice - count
- The table of sorted nods of graph.
- The area describing construction of graph (matrix of incidences)
- The area constains sorted in order of tasks by the BFS algorithm.
- The area of nods of graph.

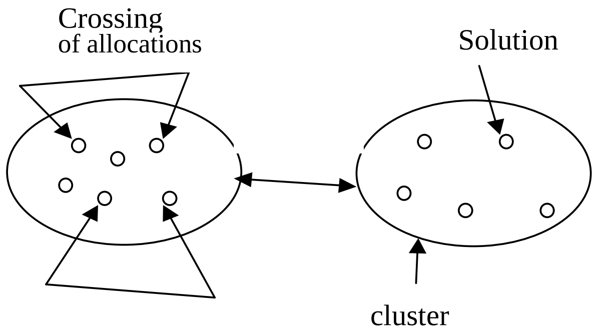


Fig. 3.2. The operations of crossing on different structures of data. Prevention of formation of defective solutions.

3.1.7 *The structure "Nod of graph of tasks"*

The structre describing the nod in graph:

- Number of nod.
- Level in graph.
- Predecessors' list.
- Successors' list.

3.1.8 *The structure "Resources"*

It contains information describing available resources:

- Number of processors
- Number of features describing the given processor.
- The area of structures describing the processor.

3.1.9 *The structure "Processor"*

It contains information describing processor:

- Type (universal, dedicated).
- Cost of operating memory.
- Cost of processor.
- The area of times of executed through this processor the tasks.
- The area of power average consumptions tasks.

3.1.10 *The structure "Global temperature"*

It contains information describing the global temperature of algorithm:

- Current temperature.
- Ratio of cooling.
- Step of temperature.

During working of algorithm, the temperature will diminish with function peaceably,

$$e^{-a \cdot x} \quad (2)$$

where *a* - the ratio of cooling. The workings about step the algorithm of reducing the temperature the argument *x* be reduced in time.

3.1.11 The Structure ”Task scheduling”

It contains the functions for scheduling of the tasks:

- Initial scheduling, ASAP algorithm.
- Mutation of schedule.
- Crossing of schedule.
- Function for the counting schedule length.

3.1.12 Struktura ”Hiperspher the features of system”

It contains the information the relating similarities of features of processors and tasks.

- Co-ordinates of centre hiperspher.
- Length of diameter
- Factors hiperplane cutting hiperspher.
- Distance of all vectors inside the hiperspher from centre.

The structure of system is represented by file of linear tables of data. During it crossing it comes to exchange of data among tables. It unfortunately, many problems were not it been possible was to describe with the help of the one dimension of series of data. Linear order usually forces upon on optimized data [8], [13].

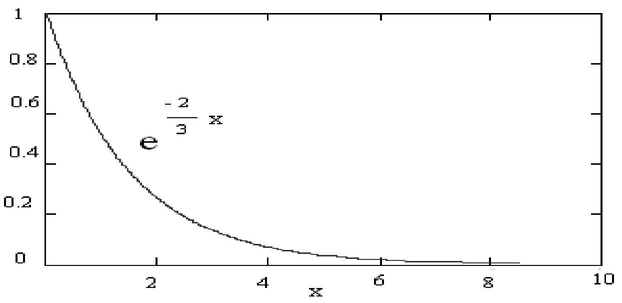


Fig. 3.3. Chart of reducing the temperature of algorithm

The evolutionary algorithm, to he could act skilfully, need in of data representing the solution structure the behaviours of lokalności. The exchange of data among individuals (the crossing) she should separate the information the describing more similar features of architecture more more seldom the than information the describing entirely different features [19]. Small changes in genotype should answer in solution which genotype represents small changes.

Putting on linear order on multidimensional information, wears out lokalność becomes. This problem the structure of data representing hipersferę in aim of solution was applied.

The multidimensional information becomes recorded in figure of vector. We interpret as reverse of local distance n - dimension vectors inside n - dimension hiperspher.

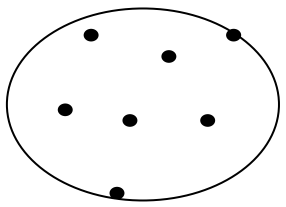


Fig. 3.4. Two dimension hipersfera (circle). Resources be described by two features here, e.g. time and cost.

The furthest distant from me vectors mark diameters and centre hyper sphere.

Algorithm keeps two hyper sphere:

- Task hyper sphere - two the dimensional, representing task graph structure. Each of the nods is defined by two coordinates: an indicator obtained through topological sorting (the tasks are “closest” if one of them is adirect successor of the other), and an indicator calculated from the BFS algorithm parallel tasks are equally distant from the beginning of the graph).
- Resources hyper sphere is three-dimensiona representing the depedencies of resource features. Each of the resources may be defined by the following coordinates: cost, speed and power consumption.

3.2 Partition of resources

It is the data the graph of tasks, pool of resources as well as criterions of optimality. The algorithm of partition of resources has determine resources, which have execute all tasks with all criterions.

3.2.1 Initial of algorithm

The aim initial of algorithm is of the construction of architecture of system the simplest and first. The architecture of system must base of accessible resources and realise required functions and set criterions. Algorithm executes following steps:

3.2.1.1 Construction of graph of tasks

On basis of input data the structure the describing graph of tasks is built. The graph of tasks represents the functionality of system. After creation of graph, nods be sorted. The topological order defines the position of tasks in graph. Equivalent levels become for nodes in graph. This features of tasks are used in scheduling algorithm. If tasks will be scheduling according to levels in graph then it will assure throwed of order constarints. The levels of tasks on following graph were marked.

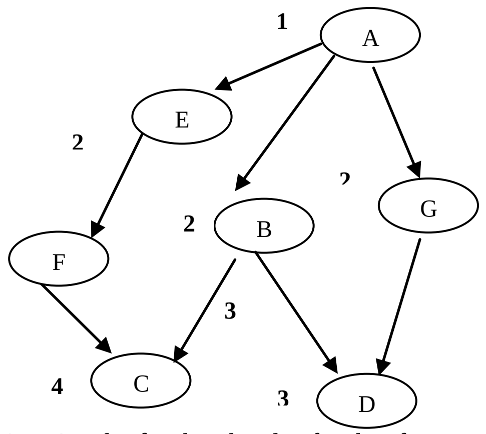


Fig. 3.5. Graph of tasks - levels of tasks after topological sorting

The searching of graph is for partition of resources the next step with the help of the algorithm the BFS [12]. The nodes of graph are assigned equivalent indices. Indices these keep the information relating the is parallel of tasks in graph. The following drawing represents the levels of tasks formed in result searching the graph with assistance of BFS.

3.2.1.2 Creation resources

On basis of data input, the representing the accessible resources object be built (the processors). Object this throws open the relating the resources information (the cost, the cost of operating memory, times of executing on processors the tasks, averages the power consumptions, the relative speed of processor, power relative consumption).

3.2.1.3 Creation population

They are the parameters of population:

- Number of clusters in population.
- Number of solutions in clusters,
- Graph of tasks - the functionality.
- Accessible resources.
- Criterion of alloy - defines the quantity of loop of algorithm when the improvement of solution did not happen.

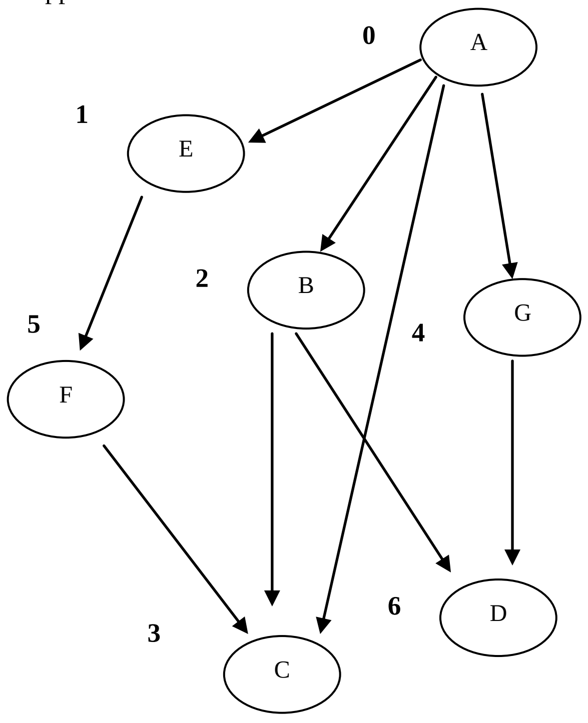


Fig. 3.6. The indices of nodes after searching the grafu by the BFS

- Map of costs - defines, which of criterions of optimization will be the taken into account during finding the optimum solution.
- Regard for universal processors the costs of operating memory during optimizing cost.
- Maxiumum number of generations of evolving solutions inside the clusters.
- Criterions of optimization - the maximum number of processors, maximum cost, maximum time, power maximum consumption.
- Size the best individuals' letters. Nodominated and the best individuals to this list be recorded. They longer list this algorithm can remember suddenly more individuals. List behaves how queue FIFO.

The created objects of clusters and solutions in clusters, and also objects hypersphere: processors and graph of tasks. The created also the object of global temperature of algorithm. The global algorithm "temperature" is initialized at this stage as well.

3.2.1.4 Initialization hypersphere

Two hypersphere be created: processors and graph of tasks. The hypersphere of processors has since 1 to 3 dimensions. The dimension depends from number of optimized features (cost, time, power consumption). Hypersphere for the graph of tasks is two dimension.

3.2.1.5 Definitions of hyper sphere

- 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.

3.2.1.6 Population initialization

- Clusters and solutions are initialized randomly.
- For every 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 they are allocated.

3.2.1.7 Initializing the allocation of tasks to resources

- A vector of resources for allocation is taken for each task.
- Resource type and number are randomly assigned to the tasks.
- Task scheduling by the ASAP (As Soon As Possible) algorithm is initialized - eliminates the violations of sequence limitations.

3.2.1.8 Solution evaluation

- The following are calculated: resource cost, task completion time and power consumption; the cost is the sum of allocated resources' costs, the time of completed tasks is the time of completing the tasks on all allocated resources, power consumption is the sum of power inputs taken by selected resources.
- If for an individual representing a solution any of the optimized criteria exceeds the maximum value acceptable, the individual is punished and the survival chances of a punished individual diminish considerably.
- As the result of the above operations, we obtain a vector containing the values of optimized criteria (time, cost, power consumption).
- A solution ranking is determined (the rating of a given solution is the number of solutions in a population which do not dominate the solution).
- A solution is dominated if each of its costs is lesser from or equal to the costs of the dominant solution (for optimization in the Pareto sense) [5].

3.2.1.9 Cluster evaluation

The solution cluster ranking is created. The rating of a cluster is the sum of the ratings of all solutions within the cluster.

3.2.2 Resource selection

The input data for resource selection are the task graph, the library of available resources and the optimization criteria, and its goal is to partition tasks into the software and the hardware part and to select resources for the realization of all tasks consistent with the established optimization criteria. The diagram of the algorithm of resource selection is showed on Fig. 3.7.

3.2.2.1 *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.

3.2.2.2 Genetic operators

The cluster mutation operator consists in mutating allocation vectors 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 global temperature, the resource is added, otherwise it is subtracted. Adding resources is limited by the maximum resource number parameter. At the beginning of the algorithm operation, resources will be added to the structure. As the algorithm approaches the end of the run defined by the cooling process, resources will be subtracted. This is aimed at creating a cost-effective structure. The cluster crossing operator consists in randomly picking two clusters and copying 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. 3.8.

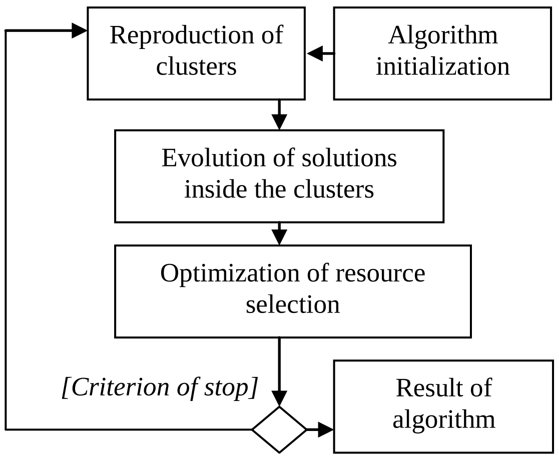


Fig. 3.7. Algorithm of resource selection

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

- Determining the cutting hyper plane by picking n points inside an n-dimensional hyper sphere.
- Creating a random permutation, e.g. for n = 3, the permutation can be (2, 1, 3).
- Constructing the point displacement vector in respect to the hyper sphere center; square coordinates are picked consistent with dimension permutations, e.g. for three dimensions with the permutation (2, 1, 3): y2 = rand() % r2, x2 = rand() % (r2 – y2), z2 = rand() % (r2 – (y2 + x2)), where: r – hyper sphere radius, and (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 equations is solved.

3.2.2.4 Saving the best solutions

After solution reproduction, a new procedure is called to save the 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 were dominated by new solutions; if there are more than one solution whose all optimized criteria values are identical, only one of those solutions is saved (the "newest" one).
- If the new solutions dominated none of the ones saved in the past, the population was not improved.
- The number of non-dominated solutions that the algorithm can save is defined by an algorithm parameter.

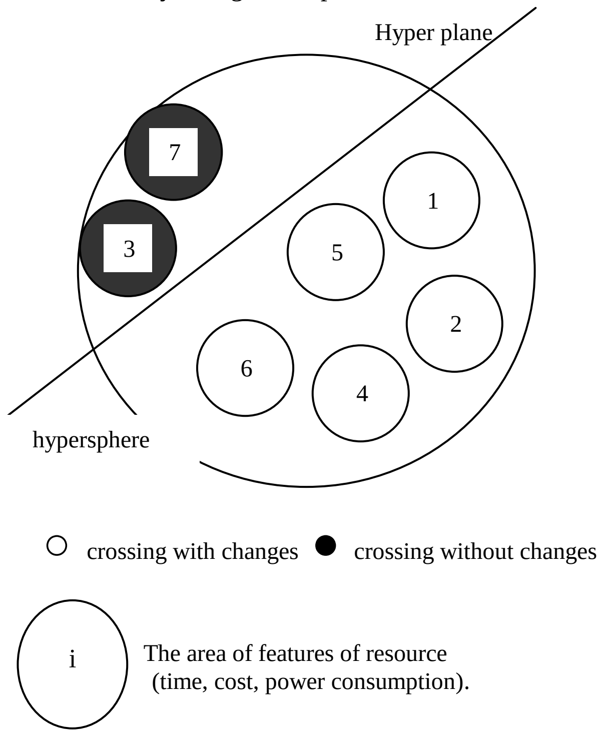

Hyper plane
7
3
1
5
2
6
4
hypersphere
○ crossing with changes
● crossing without changes
i
The area of features of resource (time, cost, power consumption).

Fig. 3.8. The crossing operator with the hyper plane

3.2.2.5 Cluster evaluation

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

3.2.2.6 Boltzman tournament

The calculations of following equation the winner of tournament be appeared on basis of result:

$$\left[1 + e^{\frac{(r1-r2)}{T}} \right] \quad (3)$$

where:

- r1 - ranking of first solution
- r2 - ranking of second solution
- T - global temperature

They are values of this function the number from compartment from < 0,1 >. We draw in aim delimitations the winner of tournament number from compartment (0,1). If she is larger from enumerated number with example then individual about ranking is winner *r1*. Second individual in opposite incident winner is (about ranking *r2*) [1].
It the analysis of results of tournament was it been possible was to conduct on basis of graph of function (Fig. 3.9.):

$$\left(1 + e^{x}\right)^{-1} \quad (4)$$

where:

$$x = \frac{-(r1 - r2)}{T} \quad (5)$$

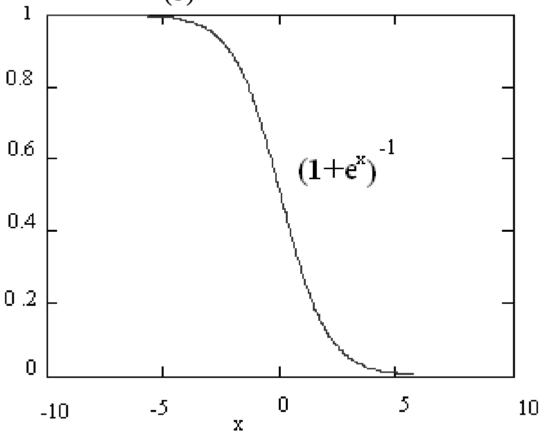


Fig. 3.9. The chart of probability of victory Boltzman tournament in dependence from global temperatur

If *r1 < r2* this *x* is negative and for high temperature larger probability exists ,that individual about rank *r1 will* win tournament than for lower temperatures. For low temperatures winner the most often will be individual about rank *r2*.
If *r1 > r2* this *x* is positive and for high temperature larger probability exists ,that individual *r2 will* win tournament than for lower temperatures. For low temperatures winner the most often will be individual about rank *r1*.

3.2.2.7 Report of algorithm

If the quantity of generations individuals' improvement during which did not happen, crosses the broadcast in criterion of alloy quantity, algorithm finishes his working. The dominated osobniki in scale of whole evolution become considered in report.

3.3 Scheduling of tasks

Task scheduling is aimed at minimizing the schedule length (the total tasks completion time).

3.3.1 Algorithm initialization

The scheduling algorithm initialization resembles the initialization of resource selection algorithm. The difference is that there is solely one cluster in which solutions evolve. The cluster allocation remains unchanged during the algorithm's run because all the resources are known for the task scheduling algorithm.

3.3.2 Algorithm of task scheduling

The diagram of the algorithm of task scheduling is showed on Fig. 3.10.

3.3.3 Solution reproduction

Solutions are reproduced using the genetic operators: crossing and mutation. Solutions are reproduced until their number doubles (the number of new solutions has been chosen arbitrarily).
The mutation operator produces one and the crossing operator two new solutions. The likelihood of using either of the genetic operators is defined by the algorithm parameters.

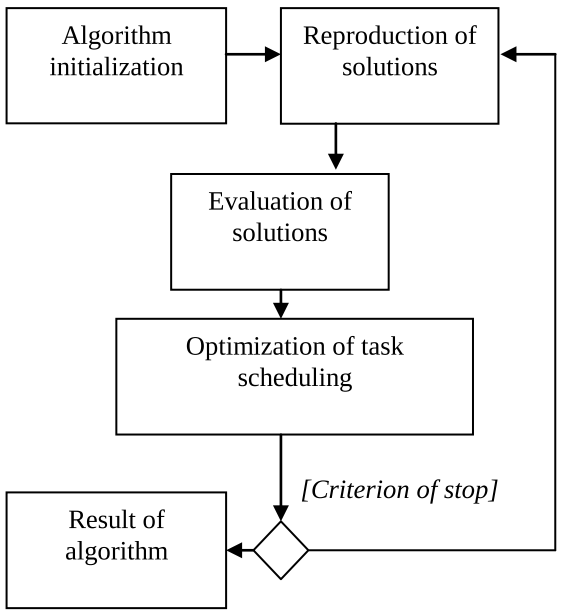


Fig. 3.10. Algorithm of task scheduling

3.3.4 Genetic operators

- **The mutation operator of task allocation to resources** acts in the following manner: a solution is randomly selected and copied. Then, the number of tasks in the system is multiplied by the global temperature. When the global temperature is high, the number of tasks changed in the allocation to resources will be greater than that in later stages of the evolution. Tasks are picked at random and allocated to resources.
- **The schedule mutation operator** acts in the following manner: if due to the mutation operation of task allocation to resources, the resource the task had been running on was changed, then the task is removed from the schedule for the "old" resource and boundaries are set on the new resource schedule between which the task may be

allocated. A location within the boundaries is picked and the task is allocated.

- **The crossing operator of task allocation to resources** resembles cluster crossing, however, the task graph hyper sphere is used for that purpose.
- **Schedule crossing operator** acts in the following way – after the allocations have been crossed, a map is created defining which parent a given feature of an offspring comes from. The offspring stores the allocation vector (obtained after crossing task allocations to resources) and the empty vector of lists with schedules of tasks on available resources. The algorithm analyzes the tasks by checking their position on the graph. For all tasks in one position, the resources on which the tasks will be performed (defined by the vector of allocation to resources) are put on the list. If in a position there is only one task ran on a given resource, the task is entered into the resource schedule, otherwise the tasks are sorted according to the starting time they had in the parent and are placed in the schedule in ascending order.

3.3.5 Solution evaluation, saving the best solutions and solution elimination

They are the same algorithms which were employed in the resource distribution algorithm. Analogical solutions are eliminated using Boltzmann tournament selection strategy [1].

3.3.6 Algorithm report

If within the number of generations determined by the annealing criterion a better individual did not appear, the evolution is stopped and the evolution report is created. The result of the algorithm operation is a set of non-dominated individuals (in the scale of the whole calculation process).

3.4 Coherent resource partition and task scheduling

The diagram of the algorithm of the coherent resources selection and tasks scheduling according to genetic approach, is showed on Fig. 3.11. The initialization of the coherent synthesis algorithm resembles the initialization of resource selection algorithm. The input parameters are the number of clusters in the population and the number of solutions in clusters. Solution clusters represent the structures sharing the same resource allocation, but with different task allocation to resources and different schedules.
The outer loop of the algorithm (realizes resource selection) is ran until the number of generations without population improvement is exceeded. This value is defined by the annealing criterion parameter. There are few outer loops at the beginning of the algorithm operation.
The number of iteration of internal loop algorithm be definite (Fig. 3.12.):

$$f(x) := -k\cdot\left(e^{-a\cdot x}\right)^3 + k \qquad (6)$$

where the k - the parameter of algorithm,
a - the annealing parameter.
Argument *x* with < 0, n >, he in every generation be enlarged about step of temperature.
N - value near which temperature is levels 0.00001.

Their number grows until it reaches the value of k with the falling of the temperature. Fewer task allocations and scheduling processes are performed at the beginning. When the temperature falls sufficiently low, each inner loop has k iterations. The number of iterations may be regulated with the temperature step parameter. The greater the step, the faster the number of inner iterations reaches the *k* value.

3.5 Computational experiments

3.5.1 The comparison coherent and non coherent synthesis with genetic algorithm. The results for tasks of dependent and nonpreemptive without cost of operating memory.

We present the analytical results obtained by testing the presented algorithms. In the tests represented by the tables and flowcharts below, we compared the results from the incoherent and coherent synthesis.

3.5.1.1 Minimize of cost

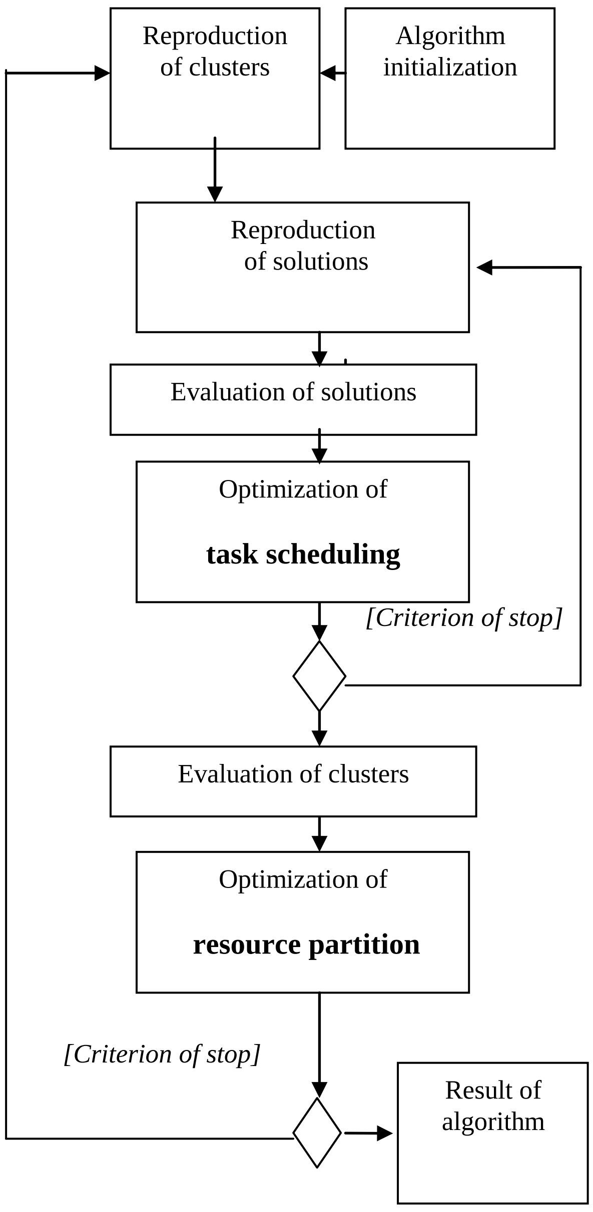


Fig. 3.11. The coherent synthesis of computer system – genetic approach

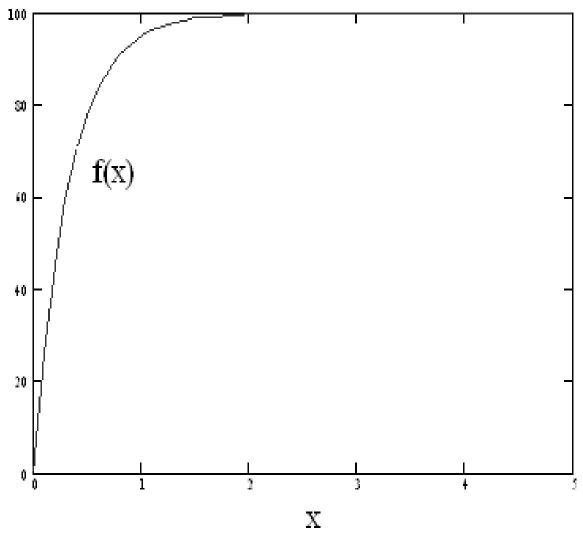


Fig. 3.12. Graph f(x) for k = 100

Table 3.1.

Tasks dependent. Minimum of cost.

Number of task	Non-coherent		Coherent		
	Cost	**Time**	**Cost**	**Tim e**	Power consumption
5	1.0	5.67	1.0	5.67	**11.68**
10	1,25	7.75	1.25	7.75	**28.94**
15	1.5	8.4	1.5	8.3	**54.42**
20	1.5	11.4	1.5	8.4	**42.64**
25	1.5	14.2	1.5	14	**80.18**
30	1.5	17.6	1.5	17.5	**103.12**
35	2.5	15.75	2.5	12.5	**101.04**
40	2.5	18.25	2.5	12.1	**129.17**
45	2.5	19,5	2.5	19.4	**126.95**
50	2.75	19.4	2.75	15.9	**124.67**
55	**2.75**	**18**	**2.75**	**14.7**	**147.32**

3.5.1.2 Minimize of time

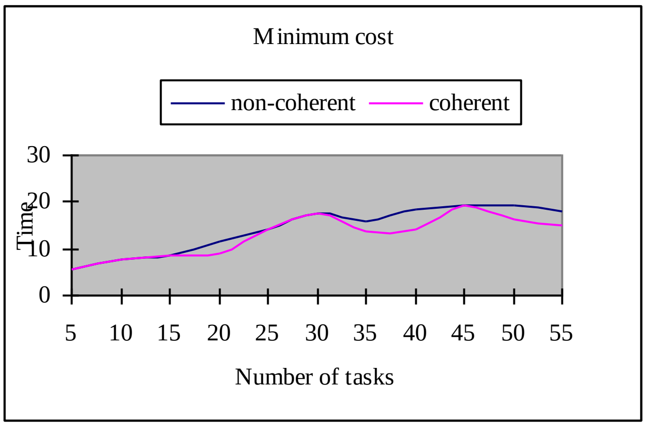
Table 3.2.

Tasks dependent. Minimum of processing time.

Number of task	Non-coherent		Coherent		
	Cost	**Time**	**Cost**	**Time**	Power consumption
5	1.00	5.57	1.8	5	**26,57**
10	1.80	7.40	1.8	7.45	**31,98**
15	3.5	7.70	3.1	6,9	**73,59**
20	3.6	8.65	3.7	7.1	**97,63**
25	4.2	7.95	3.9	7,7	**105,13**
30	4.1	9.20	2.9	7.8	**121,14**
35	5.6	8.45	5.5	7.77	**158,2**
40	7.1	8.65	6.35	7.7	**168,82**
45	8.6	8.6	5.1	5.5	**230.11**
50	8.4	8.65	7.3	7.45	**190.64**
55	**9.51**	**9.53**	**7.7**	**7.95**	**222.78**

3.5.1.3. Charts

Chart 3.1.



During cost optimization, both algorithms yielded similar cost values for all tested task sets. However, the coherent algorithm improved time optimization for graphs exceeding 30 tasks. For 50 tasks it achieved a 15% improvement of the task completion time Chart 3.1.

Chart 3.2.

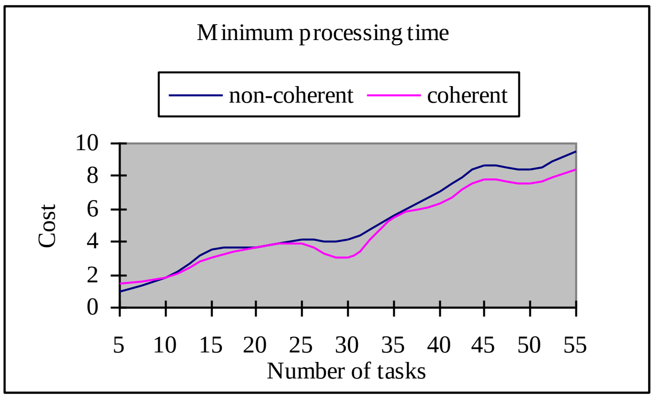
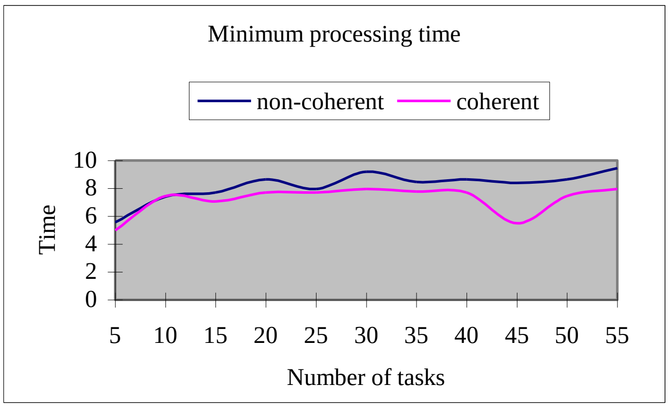


Chart 3.3.



When the flowchart reflecting the dependence of time from the number of system tasks is considered (Chart 3.2), time minimization is comparable for both algorithms. Nevertheless, once the chart showing the interdependence of cost and the number of tasks is analyzed (Charts 3.3), it is clear that the solutions yielded by the coherent algorithm are far less expensive than those from the incoherent algorithm. The coherent algorithm achieves similar task completion times in solutions much cheaper from those found by the incoherent algorithm.

3.5.2. The comparison coherent and non coherent synthesis with genetic algorithm. The results for tasks of dependent and nonpreemptive with cost of operating memory.

3.5.2.1 Minimize of cost

Table 3.3.

Tasks dependent. Minimum of cost.

Number of task	Non-coherent		Coherent		
	Cost	**Time**	**Cost**	**Time**	Power consumption
10	1.9	7.27	1.5	7.6	**35.47**
20	2.25	11.99	2	12.25	**40.17**
30	2.25	15.33	2.25	15.4	**86.39**
40	2.5	18.67	2,5	18.66	**111.1**
50	2.75	20.6	2.73	20.25	**166.87**
60	3,25	29.25	2.7	17.8	**242.29**
70	2.5	32	2.5	32	**201.29**
80	2.75	28.8	2.39	28.8	**380.28**
90	2.25	48.25	2.25	42.1	**247.93**
100	2.6	45.6	2.2	45.6	**311.85**
110	**2.6**	**56.25**	**2.2**	**50.4**	**320.98**

3.5.2.2 Minimize of time

Table 3.4.

Tasks dependent. Minimum of time.

Number of task	Non-coherent		Coherent		
	Cost	**Time**	**Cost**	**Time**	Power consumption
10	1.2	13.95	1.5	9.8	**30.77**
20	2.5	19.29	2.2	15.,95	**59.64**
30	3.67	15.45	3.3	12.25	**107.22**
40	4.4	15.85	4.4	13.45	**159.94**
50	5.5	15.8	5.1	15.05	**197.36**
60	5.6	21.45	5.7	13.45	**252.37**
70	7.3	20.15	7.7	16.40	**340.48**
80	7.6	17.45	7.2	16.3	**242.51**
90	8.5	24.45	7.7	20.15	**302.94**
100	10	19.75	7.9	18.85	**358.76**
110	**10.33**	**24.8**	**9.1**	**20.8**	**421.98**

3.5.2.3 Charts

Chart 3.4.

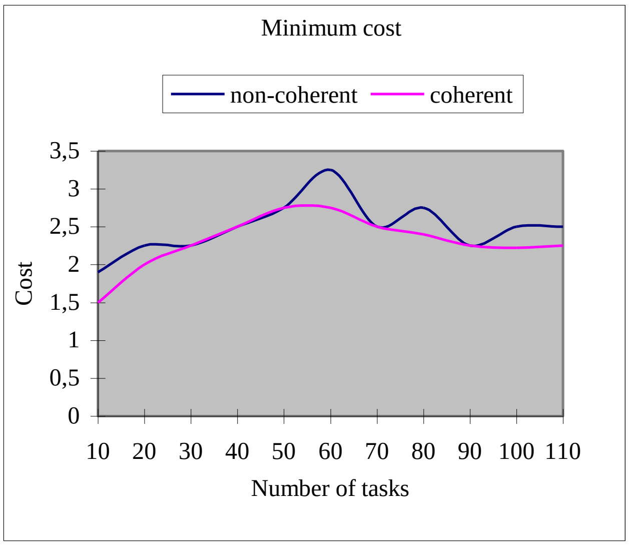


Chart 3.5.

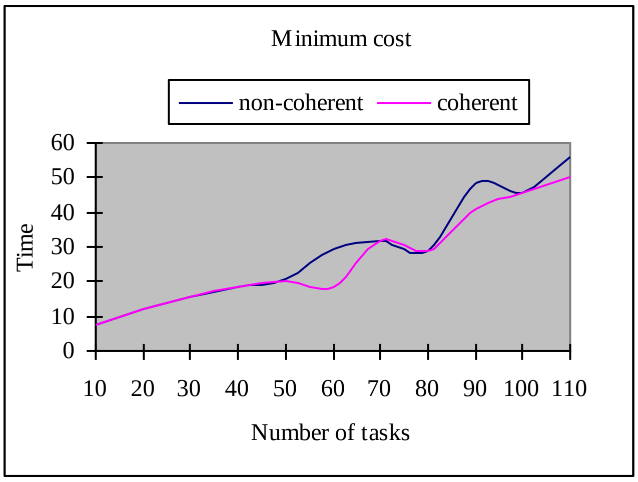


Chart 3.6.

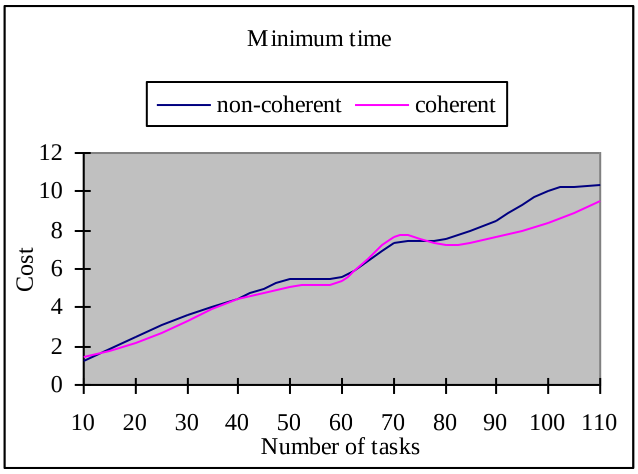
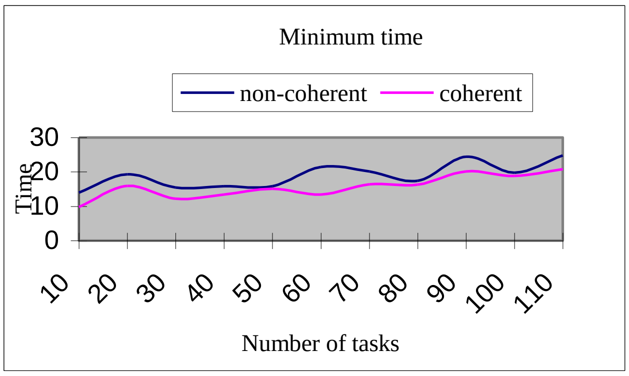


Chart 3.7.



The coherent algorithm improved time optimization. The solutions yielded by the coherent algorithm are far less expensive, as well. During minimization of time the got architectures are complex and then the scheduling algorithm has the larger possibilities for optimization. This is consequence of this that coherent algorithm gets better results than non-coherent algorithm.

3.5.3. Mult-icriterions optimization. The optimization of time of executing, power consumption and cost. Results for dependent tasks.

Tests were conducted for nonpreemptive and dependent tasks. Parameters of constraints: the maximum number of processors - 5, maximum cost - 3, maximum time 25. Optymalizowane simultaneously. It the area of optimum solutions in result was received was in sense Pareto [166]. The following tables presented solutions (in Pareto area) for the cost, time and power consumption and solution "compromissing". Searching space of solutions be led to time when global temperature reached value 0.01.

3.5.3.1 Minimize of cost

Tab. 3.5.
Multi-criterions optimization. Minimum of cost.

Number of tasks	Coherent synthesis		
	Cost	**Time**	Power consumption
5	0.5	17	**6.47**
10	0.75	15.5	**15.6**
15	1.5	8.4	**54.42**
20	1	19	**42.64**
25	2	15.75	**48.51**
30	2.25	18.4	**70.51**
35	1.5	20.8	**114.05**
40	2.75	17.75	**104.68**
45	2.25	24.67	**102.02**
50	2.25	24.25	**108.48**
55	**2.5**	**25**	**164.58**

3.5.3.2 Minimize of time

Tab. 3.6.
Multi-criterions optimization. Minimum of time.

Number of tasks	Coherent synthesis		
	Cost	**Time**	Power consumption
5	1.75	4.25	**9.56**
10	3	3.6	**35.47**
15	2.75	4.2	**77.69**
20	1.75	12.33	**37.21**
25	2	12.25	**52.24**
30	2.25	14.9	**92.18**
35	2.75	10.4	**173.83**
40	2.75	12.6	**203.57**
45	2.75	14.8	**230.11**
50	2.75	16.3	**242.29**
55	**2.75**	**18**	**268.59**

3.5.3.3 Minimize of power consumption

Tab. 3.7.
Multi-criterions optimization. Minimum of power consumption.

Number of tasks	Coherent synthesis		
	Cost	**Time**	Power consumption
5	1.75	14.75	**6.28**
10	2.5	23	**12.57**
15	2.95	18.5	**20.9**
20	1.75	21	**28.78**
25	2.5	23	**40.46**
30	2.75	24.6	**54.73**
35	3	13.33	**78.3**
40	2.75	15.85	**112.03**
45	2.25	24.67	**95.44**
50	2.25	24.5	**105.99**
55	**2.5**	**25**	**164.58**

3.5.3.4 Compromissing solution

Tab. 3.8.
Multi-criterions optimization. Compromissing solution.

Number of tasks	Coherent synthesis		
	Cost	**Time**	Power consumption
5	1.75	6.75	**9.26**
10	1,5	6.2	**35.47**
15	2.95	15.5	**24.01**
20	1.75	12.83	**35.45**

25	2	14.5	**51.25**
30	2.75	16.9	**63.58**
35	2	18	**78.3**
40	2.75	17.75	**104.68**
45	2.25	21.75	**99.5**
50	2.25	23.88	**113.26**
55	**2.5**	**25**	**164.58**

3.5.3.5 Charts

Chart. 3.8.

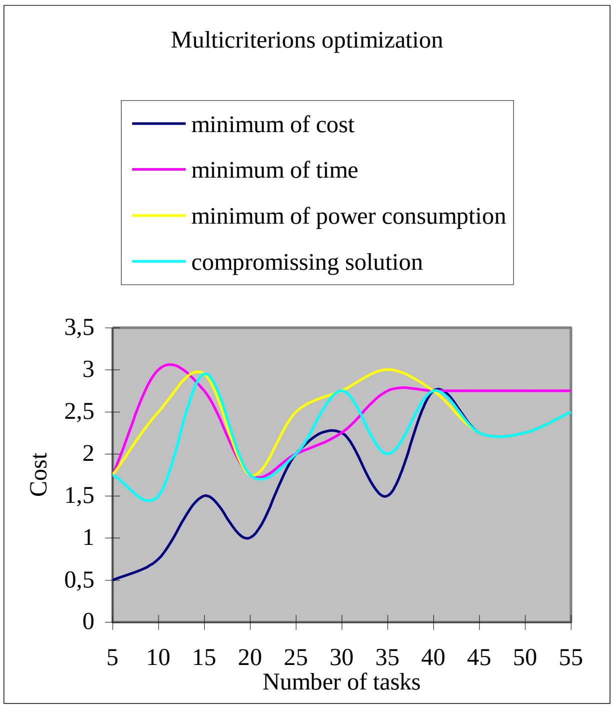


Chart. 3.9.

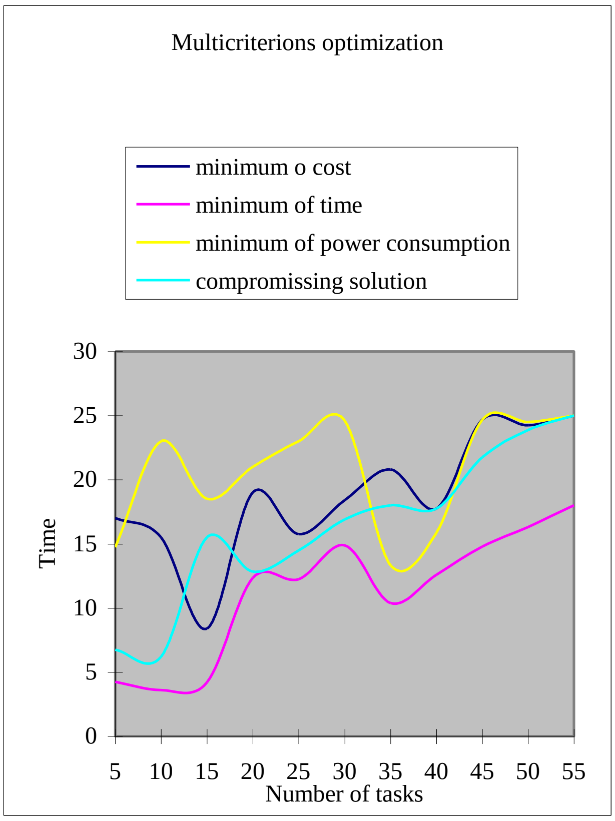
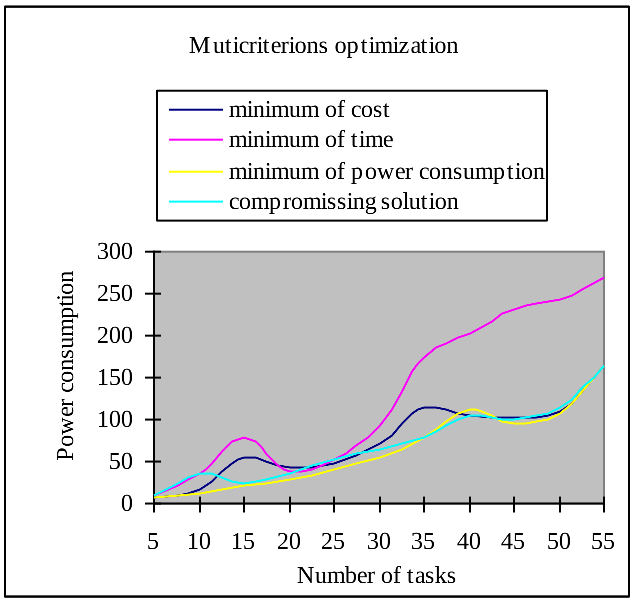


Chart. 3.10.



3.5.4. *Multi-criterions optimization. The optimization of time of executing, power consumption and cost. Results for dependent tasks with cost of operating memory.*

Tests were conducted for nonpreemptive and dependent tasks. Parameters of constraints: the maximum number of processors - 5, maximum cost - 3, maximum time 25, optimized simultaneously. It the area of optimum solutions in result was received was in sense Pareto. The following tables presented solutions (in Pareto area) for the cost, time and power consumption and solution "compromissing". Searching space of solutions be led to time when global temperature reached value 0.01.

3.5.4.1 Minimize of cost

Tab. 3.9.
Multi-criterions optimization. Minimum of cost.

Number of tasks	Coherent synthesis		
	Cost	**Time**	Power consumption
10	5.5	1.9	**56.07**
20	1.5	18,5	**33.15**
30	5.9	23	**82.41**
40	1.5	50	**81.73**
50	1.25	45.5	**137.2**
60	2.75	22	**299.61**
70	2.5	44.67	**158.2**
80	2.25	47	**179.99**
90	4.25	33.8	**311.6**
100	5.25	46	**291.7**
110	**4.25**	**47**	**431.76**

3.5.4.2 Minimize of time

Tab. 3.10.
Multi-criterions optimization. Minimum of time.

Number of tasks	Coherent synthesis		
	Cost	**Time**	Power consumption
10	6.5	1.9	**43.61**
20	2.75	7.3	**105.25**
30	5.9	23	**82.41**
40	7	20.67	**111.88**
50	4.25	13.4	**205.63**
60	4.25	16.2	**257.48**
70	2.5	32	**175.24**
80	3.25	31	**189.78**
90	4.25	22.2	**393.7**
100	5.75	20.06	**390.77**
110	**5.5**	**20.1**	**558.53**

3.5.4.3 Minimize of power consumption

Tab. 3.11.
Multi-criterions optimization. Minimum of power consumption.

Number of tasks	Coherent synthesis		
	Cost	**Time**	Power consumption
10	6.5	6.67	**23.93**
20	1.5	18,5	**33.15**
30	7.7	45	**54.44**
40	1.5	50	**81.73**
50	1.25	45.5	**137.2**
60	4.25	21.8	**225.56**
70	2.5	32	**175.24**
80	2.25	47	**179.99**
90	4.25	33.8	**311.6**
100	5.25	46	**291.7**
110	**4.3**	**49**	**429.31**

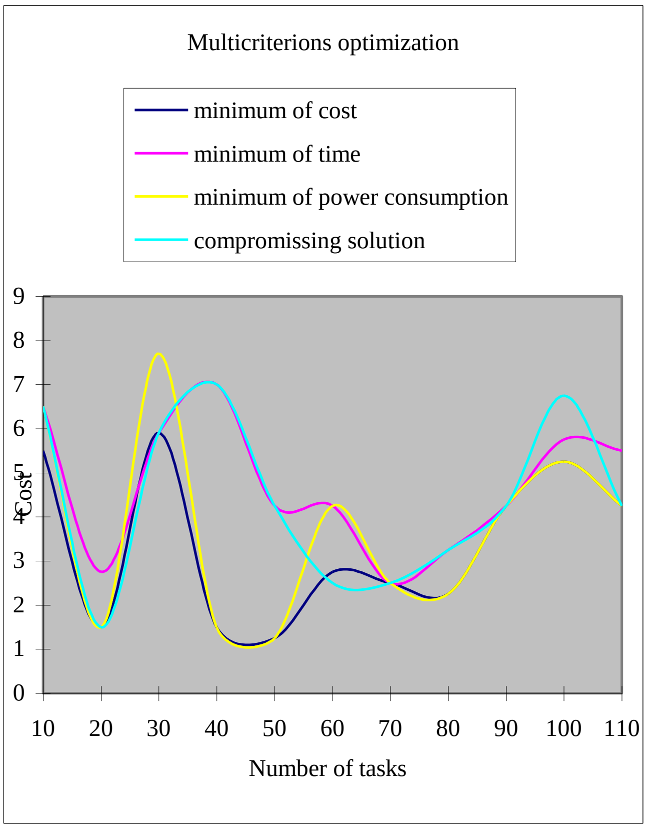
3.5.4.4 Compromissing solution

Tab. 3.12.
Multicriterions optimization. Compromisssing solution.

Number of tasks	Coherent synthesis		
	Cost	**Time**	Power consumption
10	6.5	2	**37.99**
20	1.5	18,5	**33.15**
30	5.9	23	**82.41**
40	7	23	**121.56**
50	4.25	16.2	**186.05**
60	2.5	32	**175.24**
70	2.5	38	**167.59**
80	3.25	37	**183.67**
90	4.25	28.6	**328.73**
100	6.75	30.33	**336.36**
110	**4.25**	**41.8**	**435.77**

3.5.4.5 Charts

Chart 3.11.



3.6 Conclusions

This graphs presented of multicriterions optimizationform coherent synthesis of computer system. The designer in result of working of algorithm receives in sense the gathering of optimum solutions Pareto. It stays with the designer's task the selection the most answering his requirements of solution. In dependence from this what are for system requirements it was it been possible to lean on one of got results. To to get to know for given authority of problem the specific of space of solutions well, important the use is long the list of remembering the best solutions (in tests the parameter of algorithm "it quantity *the best" it was* established was value 50). Important the settlement of slow refreshing the algorithm is equally (the parameters "the *step of temperature"* 0.1 and "the *coefficient of cooling*" - in dependent on from quantity of tasks in system; generally smaller than 0,05). We prevent thanks this sale large convergence in population [79], [98]. The algorithm searches near smaller temperature, the larger area in space of solutions. It it was noticed was also that the larger probability of mutation helps the finding the better architecture of system, and the larger probability of crossing improves the optimization of temporary criterion.

Chart. 3.12.

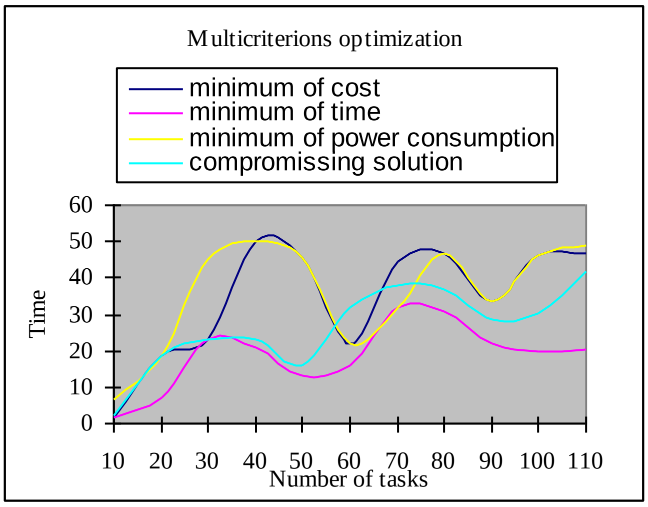
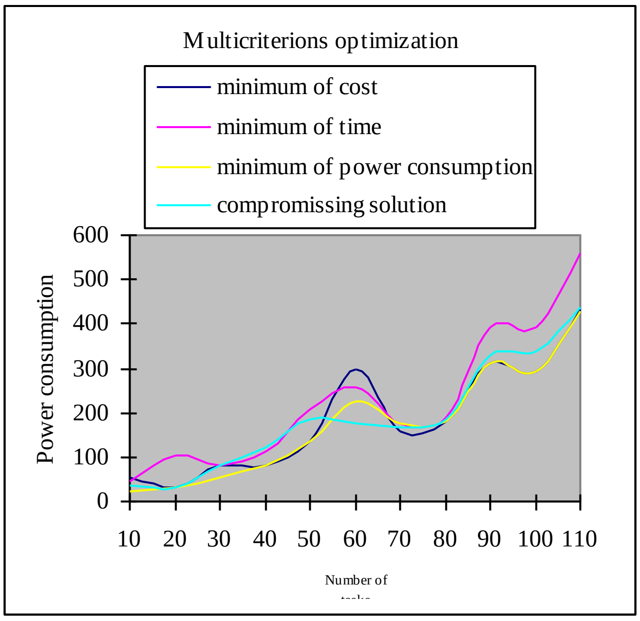


Chart 3.13.



3.7 Example coherent synthesis's applied

3.7.1 Specification of system

We assume, that algorithm of coherent synthesis has to design computer system described on this graph of tasks – Fig. 3.13. We have pools of available resources:

Tab. 3.13.

Available resources

ID	Type	Cost	Cost of memory	Power consumption
P1	universal	1	0.15	0.001
P2	universal	1.5	0.2	0.002
A1	dedicated	0.7	0	0.001
A2	dedicated	0.9	0	0.002

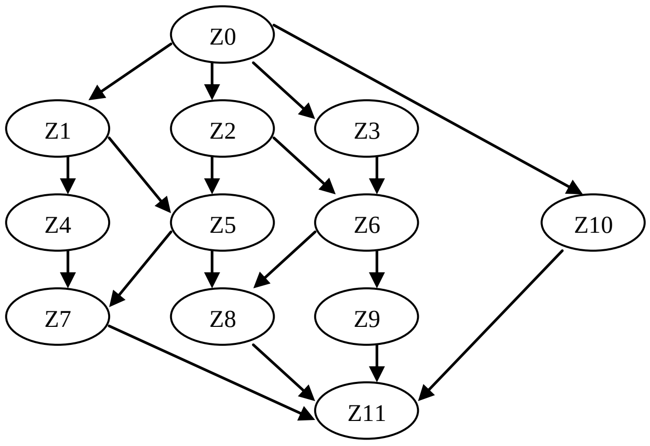


Fig. 3.13. Graph of tasks

In other requirements and constraints of system are following:

Tab. 3.14.

Times of tasks's processing.

If 0 - processor does not be adapted to realization of this task.

Tasks	Processors			
	P1	P2	A1	A2
Z0	2	1	3	2,5
Z1	4	2	1,5	0
Z2	2	1	3.5	3
Z3	4	2	0	0
Z4	5	2,5	0	0
Z5	6	3	2	1
Z6	3	1,5	0	0
Z7	2	1	0	0
Z8	1	0,5	0	0
Z9	4	2	0	1
Z10	2	1	0,5	0,4
Z11	2	1	0,5	0,4

Tab. 3.15.
Power consumption of tasks's processing.
If 0 - processor does not be adapted to realization of this task.

Tasks	Processors			
	P1	P2	A1	A2
Z0	1	1,5	1,5	1,5
Z1	2	3	1	0
Z2	1	1,5	2	2
Z3	2	3	0	0
Z4	2,5	3,5	0	0
Z5	3	4,5	2	1,5
Z6	1,5	3	0	0
Z7	1	2	0	0
Z8	0,5	1	0	0
Z9	2	3	0	1,5
Z10	1	2	0,7	0,6
Z11	1	2	0,7	0,6

Processor P2 is quicker from processor P1 but processor P2 takes more power and is dearer than P1. Dedicated processor A1 can realize tasks Z0, Z1, Z2, Z5, Z10 and Z11 and he is adequate for tasks: Z5, Z10 and Z11. Task Z0 and Z2 can be executed on this processor but time of their realization is longer than on universal processors. This processor is cheaper from dedicated processor A2, but the power consumption has greater. Dedicated processor A2 is adequate for execution of tasks Z5, Z9, Z10 and Z11. This processor is insufficient for tasks Z0 and Z1 than universal processors, but more suitable than processor specialized A1. The cost of processor A2 is greater than processor A1.

3.7.2 *Results of optimization*

Simplifying of analysis we assume that four is the maximum number of processors.

3.7.2.1 *Minimize of cost (without of cost of memory)*

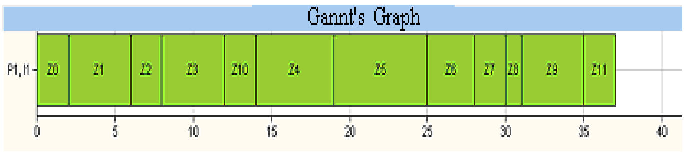


Fig. 3.14. Minimize of cost without of cost of memory

Tab. 3.16.
Minimize of cost without of cost of operating memory

Cost	Time	Power consumption
1	37	18,5

We for obvious reasons in this example have consisting system from one and the cheapest universal processor. Dedicated processors, which are cheaper possibilities of realization of all tasks have not in this system.

3.7.2.2 *Minimize of cost (with of cost of memory)*

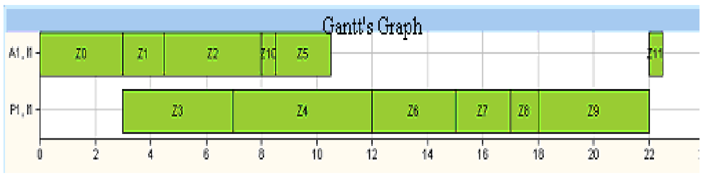


Fig. 3.15. Minimize of cost (with of cost of memory)

Tab. 3.17.
Minimize of cost (with of cost of operating memory)

Cost	Time	Power consumption
2,6	23,5	17,4

We for this example will receive system folded from one universal processor (cheaper) as well as one dedicated processor (cheaper). Here two processors more suitable is apply. Cost of realization of all tasks on the cheapest universal processor is equal 2,8 (the cost of processor and 12 the tasks the by 0,15 cost of unit of memory).

3.7.2.3 *Minimize of processing time*

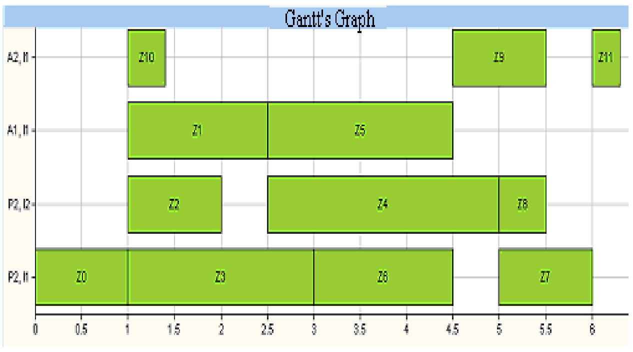


Fig. 3.16. Minimize of processing time

Tab. 3.18.
Minimize of processing time

Cost	Time	Power consumption
6	6,3	21,2

For example in which in have the minimization of time is generated system, which are two quicker universal processors and two dedicated processors A1 and A2. Algorithm counted schedule in which processors A1 and A2 executed tasks for which they are dedicated.

3.7.2.4 *Minimize of power consumption*

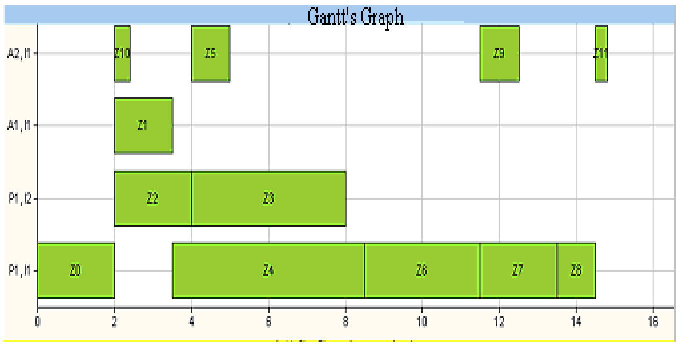


Fig. 3.17. Minimize of power consumption

Tab. 3.19.

Minimize of power consumption

Cost	Time	Power consumption
4,65	14,8	14,7

For this example algorithm generated system folded from two cheaper dedicated processors for which the power consumption is the lowest for almost all tasks. Task Z1 was executed on processor A1 because executing this task processor A1 is dedicated. Task Z10, Z5, Z9 and Z11 are executed on dedicated processor A2 on which is smallest the power consumption.

3.7.2.5 *Minimize of time and cost (for given maximum of cost)*

Maximum cost was given on value 4.

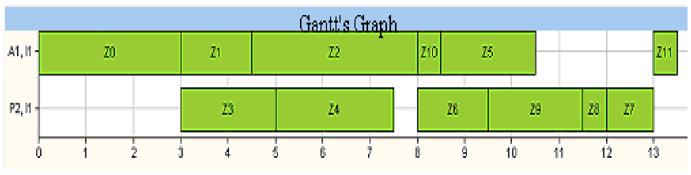
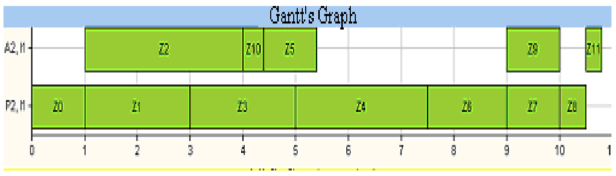


Fig. 3.18. Minimize of time and cost. Cheaper system.

Tab. 3.20.

Minimize of time and cost. Cheaper system.

Cost	Time	Power consumption
3,4	13,5	23.4



3.19. Minimize of time and cost. Quicker system.

Tab. 3.21.

Minimize of time and cost. Quicker system.

Cost	Time	Power consumption
3,8	10,8	23,2

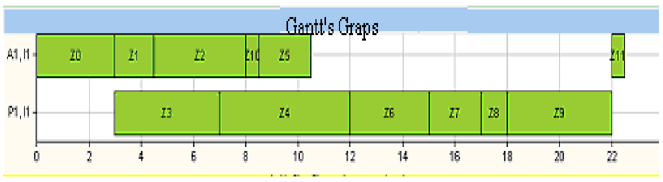


Fig. 3.20. Minimize of cost and time. The cheapest system. Improvement of time.

Tab. 3.22.

Minimize of cost and time. The cheapest system. Improvement of time.

Cost	Time	Power consumption
2,6	22,5	17,4

Algorithms for multicriterions optimization receive the area of optimum solutions in sense Pareto. First and second solution have the costs greater than settled the maximum as well as have larger time from minimum time of system. However they find among these conflicting requirements compromise. First of solutions is cheaper since second about 0,4. The time of realization of tasks is longer about 2,7. The power consumption is lower for dearer system. The solution for last case is about the smallest of cost and other criterions are simultaneously estimated. Solution this has the same cost how the cheapest system (solution for minimize of cost) and schedule length about 1 reduces simultaneously.

3.7.2.6 *Minimize of cost and power consumption (for given maximum of time)*

The maximum of time was given on value 20.

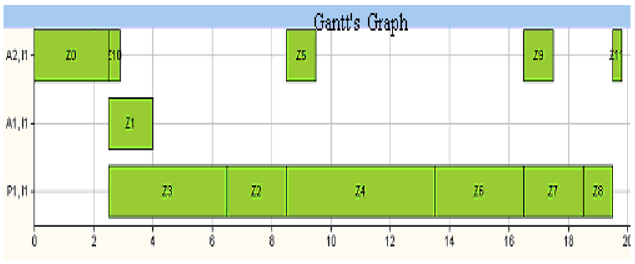


Fig. 3.21. Minimize of cost and power consumption. Maximum time equal 20. Cheaper system.

Tab. 3.23.

Minimize of cost and power consumption. Maximum time equal 20.

Cheaper system.

Cost	Time	Power consumption
3,5	19,8	15,2

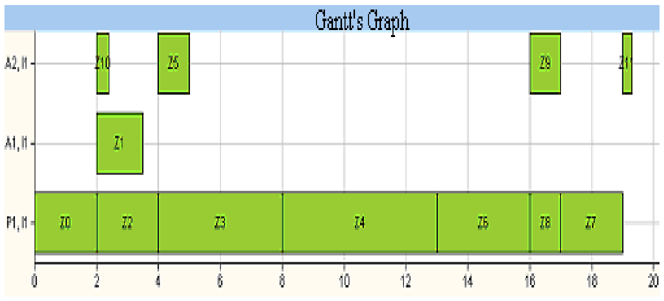


Fig. 3.22. Minimize of cost and power consumption. Maximum time equal 20. Dearer system.

Tab. 3.24.

Minimize of cost and power consumption. Maximum time equal 20.
Dearer system.

Cost	Time	Power consumption
3.65	19.3	14.7

Criterion of time was the requirement for resultant of system: he had to finish the executing the tasks before 20 units of time. Other criterions have be optimized: i.e. time and cost. Algorithm generated two optimum solutions in sense Pareto. System for first solution executed tasks to 19,8 units of time, has the larger power consumption and his cost is smaller. Systems these differ of the execution of task Z0, only. In first system this task is executed through dedicated processor, which has larger power consumption and he need not additional operating memory. Execution of this task through universal processor P1 has smaller of the power consumption and he need to execution of task the additional memory. The second solution generated system for which time executing all tasks is shorter.

3.7.2.7 Multicriterions optimalization. Coherent minimize of cost, time and power consumption (for given maximum of time).

Maximum of time was given on value 15.

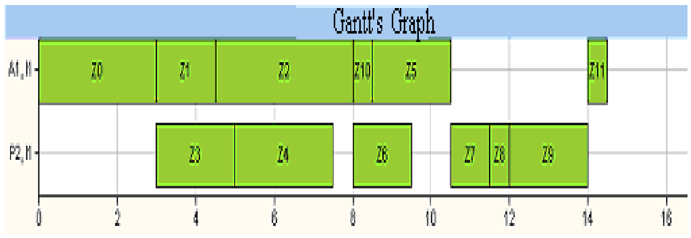


Fig. 3.23. Minimize of cost, time and power consumption. Cheaper system.

Tab. 3.25.
Minimize of cost, time and power consumption. Cheaper system.

Cost	Time	Power consumption
3,4	14,5	23,4

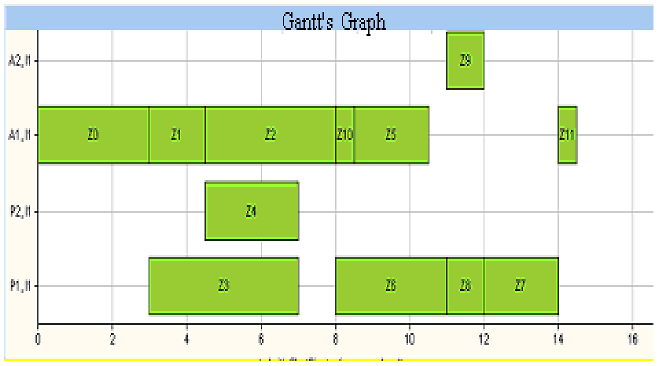


Fig. 3.24. Minimize of cost, time and power consumption. Cheaper system. System with optimized power consumption.

Tab. 3.25
Minimize of cost, time and power consumption. Cheaper system.
System with optimized power consumption.

Cost	Time	Power consumption
4,9	14,5	17,9

The algorithm generated area of optimum solutions in sense Pareto for three criterions. Generated solutions get the different compromise among conflicting criterions. Two systems to analysis were chosen. Finish execution time for all of tasks is 14,5 unit. First system consists with 2 processors and his cost is small. However executing of tasks on processor P2 causes the power consumption considerable [73]. The dedicated processor executes the tasks with smaller power consumption than processor A2. Task Z2 is except. Realization of this task on processor P2 would cause the crossing the limit of time. Second system consists with 4 processors. This system is dearer (150%) however the power consumption reduces significantly and the limit of time of realization tasks fulfils simultaneously.

4. Résumé

The paper describes genetic algorithms and their implementation flowcharts. Moreover, it presents selected results of analytical experiments for resource selection and task scheduling. The paper explores the coherent synthesis algorithm of computer systems, in which resource selection and task scheduling optimization processes are realized concurrently and coherently. The coherent approach in the synthesis generates common and interdependent solutions regarding the system structure (type and configuration of the selected resources), as well as the scheduling of tasks ran on those resources. In the presented approach, the cost of resources (system cost), the time of completing all tasks (system speed) and the power consumption of the system are optimized. The coherent algorithm yields much (up to 40%) better solutions, which is proved by analytical experiments.

References

[1] Aarts E.H.L. Korst J., "Simulated Annealing and Boltzmann Machines", J. Wiley, Chichester, 1989.
[2] D'Ambrioso J., Hu X., "Configuration Level Hardware/Software Partitioning for Real-Time Systems", Proc. of the Int. Workshop on Hardware/Software Codesign, Vol. 14, 34-41, 1994.
[3] Axelson J., "Architecture Synthesis and Partitionig of Real-Time Systems: A Camparison of Three Heuristic Search Strategies", Proc. of the Int. Workshop on Hardware/Software Codesign, 161-165, 1997.
[4] Błażewicz J., Ecker K., Pesch E., Schmidt G., Węglarz J., "Handbook on Scheduling, From Theory to Applications", Springer-Verlag Berlin Heidelberg, 2007.
[5] Błażewicz J., Ecker K., Plateau B., Trystram D., "Handbook on Parallel and Distributed Processing", Spinger-Verlag, Berlin – Heidelberg, 2000.
[6] Brucker P., Knust S., "Complex Scheduling", Springer, 2006.
[7] Coffman E. G., Jr., "Computer and Job-shop scheduling theory", John Wiley&Sons, Inc. New York, 1976.
[8] 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.
[9] Drabowski M., M., Rola M, Rościcki A., "Algorytmy neuronowe w sterowaniu rozdziałem zadań i zasobów w kompleksie operacji", 2rd International Congress of Intelligent Building Systems, INBus2002, Kraków, 45-52, 2002.
[10] Drabowski M., Czajkowski K., "Task scheduling in coherent, co-synthesis of computer system", Advanced Computer Systems – Computer Information Systems and Industrial Management Application (ACS-CISIM 2005), in. Image Analysis, Computer Graphics, Security Systems and Artificial Intelligence Applications, vol. 1, 53-60, 2005.
[11] Drabowski M., "Par-synthesis of multiprocessors parallel systems", International Journal of Computer Science and Network Security, Vol. 8, No. 10, 90-96, 2008.
[12] Eiben A.E. Aarts E.h.l. van Hee K.H., "Global convergence of genetic algorithms: A Markov chain analysis", LNCS 496, 4-9, 1991.
[13] Gajski D.D., Dutt N.D., Wu A.C.H., Lin S.Y., "High-level Synthesis. Introduction to Chip and System Design", Kluwer Academic Pub., Boston, MA, 1994.
[14] Gajski D., "Principles of Digital Design", Prentice Hall, Upper Saddle River, NJ, 1997.
[15] Garey M. R., Johnson D. S., "Computers and intractability: A guide to the theory of NP-completeness", San Francisco, Freeman, 1979.
[16] Gupta R.K., De Micheli G., "Hardware-Software Co-synthesis for Digital Systems", IEEE Design&Test of Computers, Vol. 10, No. 3, 29-41, 1993.
[17] Harel D., *Statecharts*: "A Visual Formalism for Complex Systems", Science of Computer Programming, Vol. 8, No. 3, 231-274, 1987.
[18] Henkel J., Ernst R., "High-Level Estimation Techniques for Usage in Hardware/Software Co-Design", Proc. of the Asia and South Pacific Automation Conference, 353-360, 1998.
[19] de Micheli G., "Computer-Aided hardware-Software Codesign", IEEE Micro, Vol. 14, No. 4, pp. 10-24, 1994.
[20] Schulz S., Rozenbilt J.W., Mrva M., Buchenrieder K., "Model-Based Codesign", IEEE Computer, Vol. 31, No. 8, 60-67, 1998.
[21] Sgroi M., Lavagno L., Sangiovanni-Vincentelli A., "Formal Models for Embedded System Design", IEEE Design&Test of Computers, Vol. 17, No. 2, 14-27, 2000.
[22] Steinhausen U., "System Synthesis Using Hardware/Software Codesign", Proc. of the Int. Workshop on Hardware-Software Co-Design, 1993.
[23] Teich J., Blickle T., Thiele L., "An Evolutionary Approach to System-Level Synthesis", Proc. of the Int. Workshop on Hardware/Software Codesign, 167-172, 1997.
[24] Węglarz J., "Project Scheduling – Recent Models, Algorithms and Applications", Kluwer Academic Publ., 1999.
[25] Węglarz J., "Sterowanie w systemach typu kompleks operacji", PWN, PAN, Oddział w Poznaniu, 1981.

Author Biographies

Mieczyslaw Drabowski, Assistant Professor of Department of Computer Engineering, 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 International Association for Development of the Information Society (IADIS), International Association of Science and Technology for Development (IASTED) on Artificial Intelligence and Soft Computing, International Journal of Computer Science and Emerging Technologies.
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 80 papers in major professional journals and conference proceedings.
Dr. Drabowski is a member of the council of the Polish Information Processing Society.