Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2012, Article ID 625861, 12 pages
doi:10.1155/2012/625861

*Research Article*

# Optimization of Resource Control for Transitions in Complex Systems

**Florin Pop**

*Faculty of Automatic Control and Computers, University Politehnica of Bucharest, Splaiul Independentei 313, 060042 Bucharest, Romania*

Correspondence should be addressed to Florin Pop, florin.pop@cs.pub.ro

Received 7 March 2012; Accepted 2 April 2012

Academic Editor: Cristian Toma



In complex systems like Large-Scale Distributed Systems (LSDSs) the optimization of resource control is an open issue. The large number of resources and multicriteria optimization requirements make the optimization problem a complex one. The importance of resource control increases with the need of use for industrial process and manufacturing, being a key solution for QoS assuring. This paper presents different solutions for multiobjective decentralized control models for tasks assignment in LSDS. The transaction in real-time complex system is modeled in simulation by tasks which will be scheduled and executed in a distributed system, so a set of specifications and requirements are known. The paper presents a critical analysis of existing solutions and focuses on a genetic-based algorithm for optimization. The contribution of the algorithm is the fitness function that includes multiobjective criteria for optimization in different way. Several experimental scenarios, modeled using simulation, were considered to offer a support for analysis of near-optimal solution for resource selection.

## 1. Introduction

The resources control in complex systems requires information about resources and tasks. A near-optimal assignment could be made based on some criterion function, such as minimum execution time or load balancing. There has been a steadily increasing interest, supported by advanced technological and economic developments, into dealing with very complex (dynamical) systems describing natural phenomena or manufacturing processes [1]. The particular interest in the study of complex systems is tipping points where one observes a sudden change in the dynamics, sometimes referred to as critical transitions, modeled as tasks to be submitted for execution on physical resources. For instance, medical conditions such as asthma attacks and epileptic seizures can change quickly from regular to irregular

behavior, the financial markets are known to suddenly break trends in a crisis, and climate conditions and ecological environments can change rather abruptly. The understanding of the dynamical behavior near tipping points would enable human interaction to attenuate or control the consequences of critical transitions [2].

The scheduling process is considered to be the core of resources control in complex systems. Due to the NP-complete nature of scheduling algorithms, current research directions are focused on finding suboptimal (near-optimal) solutions, which can be further divided into the following two general categories: approximate and heuristic algorithms. At global level, two-phase scheduling solution comprised of a set of heuristic subalgorithms to achieve optimized scheduling performance over the scope of overall resources is a new research subject in the present [3, 4].

Today, engineers face an increasing challenge in advanced applications with different requirements and constrains. Innovative developments for efficient mathematical approaches focused on approximate algorithms, heuristics-based methods, and bio-inspired models. The approximate algorithms use formal computational models, but instead of searching the entire solution space for an optimal solution, they are satisfied when a solution that is sufficiently *good* is found. In the case where a metric is available for evaluating a solution, this technique can be used to decrease the time taken to find an acceptable schedule. The factors which determine whether this approach is worthy of pursuit include [5, 6] availability of a function to evaluate a solution, the time required to evaluate a solution, the ability to judge the value of an optimal solution according to some metric, and availability of a mechanism for intelligently pruning the solution space. The paper proposes a mathematical approach for resources control based on a multicriteria optimization genetic algorithm. One of the well-known problems of genetic algorithms is that, for large solution space, the convergence time is high.

The rest of the paper is structured as follows. Section 2 describes the optimization methods. Section 3 approaches the multidimensional optimization methods for real-time system. The proposed genetic algorithm is described in Section 4. The tests conducted on the proposed algorithm (Section 5) highlight the improvement provided by this new approach not only in terms of convergence time, but also in terms of solution quality.

## 2. Optimization Methods for Decentralized Control

Optimization methods for resource control use heuristic (multiobjective) approaches. The allocation problem considers a set of $n$ tasks, $T = \{T_1, T_2, \ldots, T_n\}$, for some finite integer $n$, that models a set o transitions that will use a multiple processor system (e.g., cluster system) in which each transition can be characterized by multiple parameters: $T_i = \{a_i, t_i, C_i, r_i, \omega_i \ldots\}$, where $a_i$ is the *arrival time* (the time when the transition is produce), $t_i$ is the *execution time* (it can be estimated or calculated), $C_i$ is the *completion time* (with the following condition: $a_i + t_i \leq C_i$), $r_i$ is a *rate* $0 < r_i \leq 1$ (can be a normalized priority with $\sum_{i=1}^{n} r_i = 1$), $0 \leq \omega_i \leq 1$ is a *weight* (with the special normalization condition $\sum_{i=1}^{n} \omega_i = 1$), and we can have some other parameters which characterize the task.

In this model, a complex system has a number of $m$ resources $R = \{R_1, R_2, \ldots, R_m\}$. Each resource $R_j$ has specific characteristics like capacity, latency, memory type and space, CPU processing characteristics, and storage limitation. The most important characteristic used in the proposed model is the utilization rate $(u_j)$ that measures the processing capacity. The work$_j$ done by $R_j$ in order to process a task $T_i$ is defined as its running time multiplied

by the resource utilization rate, $\text{work}_j(T_i) = t_i u_j \leq (C_i - a_i)u_j$. A valid schedule is defined as $\text{Sched} = \{(T_i, R_j) \mid T_i \in T,\ R_j \in R\}$. Similarly, the work of set of scheduled tasks is $\text{work}(T, R) = \sum_{(T_i,R_j)\in \text{Sched}} \text{work}_j(T_i)$. Usually it is assumed that in the case of malleable tasks the work of a task cannot be decreased by spending more processors on it (preservation of work). Similarly the work of a task cannot be decreased by using virtualization.

To evaluate the efficiency o resource utilization a resource is considered to be *Active* (working) or Idle (waiting for new tasks). Efficiency $E(t)$ at time $t$ is $E(t) =$ Resource Active$(t)$/(Resource Active$(t)$ + Resource Idle$(t)$). In general we are looking for a feasible solution to scheduling problem. This is a schedule which meets all the requirements and constrains.

For optimization, there are *bottleneck objectives* and *sum objectives*. The scheduling problem considers the following objective for optimization: *maximum completion time* ($C_{\max} = \max_i\{C_i\}$), *weighted completion time* ($C_w = \sum_{i=1}^{n} w_i C_i$), or *maximum lateness* ($L_{\max} = \max_i\{|(C_i - a_i) - t_i|\}$).

Another important aspect of scheduling optimization considers real-time systems. These type of systems are defined as those systems in which the correctness of the system depends not only on the logical result of computation, but also on the time at which the results are produced. If the timing constraints of the system are not met, system failure is said to have occurred. Hence, it is essential that the timing constraints of the system are guaranteed to be met.

## 2.1. Heuristics for Resources Control

*Opportunistic Load Balancing (OLB)*

The Opportunistic Load Balancing heuristic selects the task $T_i$ arbitrarily from the group of tasks and assigns it to the next resource that is expected to be available [7, 8]. It does not consider the $\text{work}_j(T_i)$, which may lead to very high $C_{\max}$. If all tasks are scheduled with respect to condition $a_1 < a_2 < \cdots < a_n$ the heuristic is called *First Come First Served* (FCFS) [9].

*Minimum Execution Time (MET)*

The heuristic assigns each task selected arbitrarily to the machine with the least expected execution time for that task [10].

*Minimum Completion Time (MCT)*

The heuristic assigns each task selected in arbitrary order to the machine with the minimum expected completion time for that task [10]. The MCT combines the benefits of OLB and MET and tries to avoid the circumstances in which OLB and MET perform poorly.

*Min-Min*

The heuristic begins with the set $T$ of all task to be unscheduled. Then, the set $C$ of minimum possible completion times of all tasks on any of the machines is computed: $C_{\min} = \min_i\{C_i\}$. The task with the $C_{\min}$ is then assigned on a processor with minimum expected work.

*Max-Min*

The heuristic is very similar to min-min, but considers $C_{\max}$ and then assigns the task on a processor with minimum expected work [10, 11].

*Duplex*

The heuristic is a combination of the min-min and max-min heuristics. The heuristic performs both of the min-min and max-min heuristics and used the better solution [8, 10, 11].

*Genetic Algorithms (GAs)*

They are used for searching large solution spaces with multiple possible schedule of tasks. Each possible schedule is modeled by a chromosome that has a fitness value, which is the result of an objective function designed in accordance with the performance criteria of the problem ($C_{\max}$ or $L_{\max}$) [12].

*Simulated Annealing (SA)*

It is an iterative technique that considers only one possible solution (mapping) for each task at a time [13]. This solution is modeled like in a GA. SA uses a procedure that probabilistically allows poorer solutions to be accepted to attempt to obtain a better search of the solution space.

*A**

Heuristic is a search technique that has been applied in various task allocation problems. The $A^*$ heuristic begins at a root node that is a null solution. As the tree grows, nodes represent partial schedule. A pruning process is performed to limit the maximum number of active resources at any one time. A cost function $f(\text{Sched})$ is associated with a partial solution (e.g., $f(\text{Sched}) = C_{\max}$).

Guaranteeing timing behavior requires that the system could be predicted. Predictability means that when a task is activated it should be possible to determine its execution time with certainty. It is also desirable that the system attains a high degree of utilization while satisfying the timing constraints of the system [14–16].

## 2.2. Resource Control in Real-Time Complex System

A complex system is said to be *real-time* if there exists at least one task $T_i \in T$, which falls into one of the following categories.

(1) Task $T_i$ is a *hard real-time* task. The execution of the task $T_i$ should be completed by a given deadline, $a_i + t_i \leq C_i$.

(2) *Task $T_i$ is a soft real-time task. If a task $T_i$ finishes the work after a given deadline $C_i$, the penalty is pays. A* penalty function $P(T_i)$ is defined for the task. If $a_i + t_i \leq C_i$, the penalty function $P(T_i) = 0$. Otherwise $P(T_i) = (a_i + t_i) - C_i > 0$.

(3) Task $T_i$ is a *firm real-time* task. If a task $T_i$ finishes the work before a given deadline $C_i$, the more rewards it gains. A *reward function* $R(T_i)$ is defined for the task. If $a_i + t_i \geq C_i$, the reward function $R(T_i) = 0$ is zero. Otherwise $R(T_i) = C_i - (a_i + t_i) > 0$.

The set of real-time tasks $T$ can be a combination of hard, firm, and soft real-time tasks. Let $T_S$ be the set of all soft real-time tasks in $T$. The penalty function of the system is $P(T) = \sum_{i=1}^{|T_S|} P(T_{S,i})$. Let $T_F$ be the set of all soft real-time tasks in $T$. Similarly, the reward function of the system is $R(T) = \sum_{i=1}^{|T_F|} R(T_{F,i})$.

The following goals should be considered in scheduling a real-time system: (i) meeting the timing constraints of the system; (ii) preventing simultaneous access to shared resources and devices; (iii) attaining a high degree of utilization while satisfying the timing constraints of the system; (iv) reducing the cost of context switches caused by preemption; (v) reducing the communication cost in real-time distributed systems. In addition, the following criteria are considered in advanced real-time systems: (vi) considering a combination of hard, firm, and soft real-time activities, which implies the possibility of applying dynamic scheduling policies that respect the optimality criteria; (vii) task scheduling for a real-time system whose behavior is dynamically adaptive, reconfigurable, reflexive, and intelligent; (viii) covering reliability, security, and safety. Basically, the scheduling problem is to determine a schedule for the execution of the task so that they are all completed before the overall deadline [14, 15].

## 3. Multidimensional Optimization Methods: Applications

Multidimensional optimization methods are useful when the search space is likely to have many local optima, making it hard to locate the global optimum. In low-dimensional or constrained problems it may be enough to apply a local optimizer starting at a set of possible start points, generated either randomly or systematically (for instance, at systems locations), and choose the best result. However this approach is less likely to locate the true optimum as the ratio of volume of the search region to number of starting points increases. The application of different multidimensional optimization method proves that finding the global optimum is a hard problem.

### *Application of Simplex Method*

Scheduling of vehicles in the container terminal is often studied as a static problem in the literature, where all information, including the number of task, their arrival time, and so forth, is known beforehand. The objective is generally minimizing the total traveling and/or waiting times of the vehicles. When the situation changes, for example, new jobs arrive or a section of the terminal is blocked, new solutions are generated from scratch.

### *Application of Simulated Annealing Method*

A parallel approach of a modular simulated annealing (MSA) algorithm, a shortened SA algorithm, applied to classical job-shop scheduling (JSS) problems is presented. The JSS problems tackled are very well-known difficult benchmarks, which are considered to measure the quality of such systems.

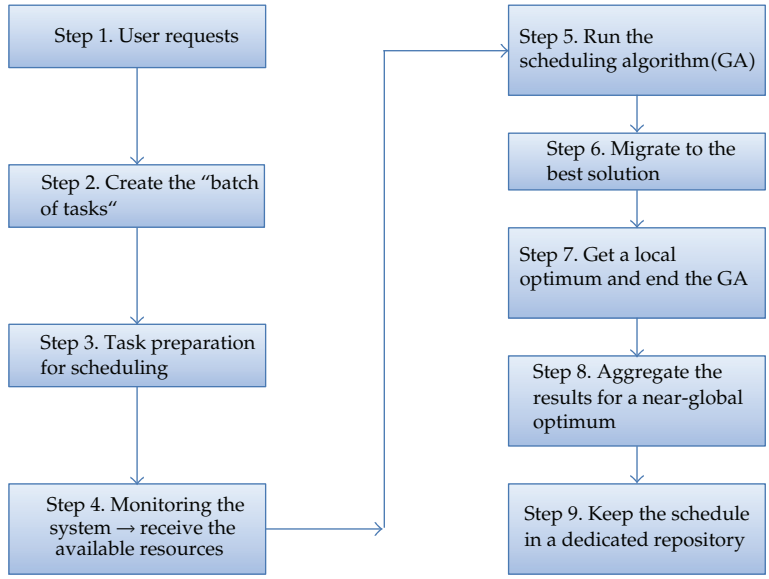


**Figure 1:** Main actions of proposed algorithm.

*Application of Genetic Algorithms*

GAs are developed for solving the machine-component grouping problem required for example a cellular manufacturing systems. GA provides a collection of satisfactory solutions for a two-objective environment (minimizing cell load variation and minimizing volume of inter cell movement), allowing the decision maker to then select the best alternative.

## 4. Genetic Algorithm for Resources Control in LDSD

In [17] Iordache et al. present a genetic algorithm for decentralized scheduling. The description of the scheduling algorithm in a logical flow of activities is described in the following steps. The important contribution of this algorithm is the fitness function that considers multiobjective criteria for optimization (see Figure 1).

*Step 1.* A user requests that one or more tasks are scheduled.

*Step 2.* The input is processed as a "batch of tasks" (group of tasks). The batch of tasks is broadcast to all the resources in the cluster.

*Step 3.* The resources receive the group of tasks to be scheduled. The tasks are inserted-sorted in a queue according to a sorting criteria like arriving time ($a_i$) or scheduling priority ($r_i$). If the number of tasks in the queue is less than a predefined length of the chromosome, they wait for $\tau$ units of time before starting the genetic algorithm. If the chromosome is still not complete at the end of the waiting period, a noninfluential padding is added. On the contrary, if the length of an arriving group of tasks exceeds the predefined dimension of the chromosome, some tasks are saved in the waiting queue and will be scheduled at the next time.

*Step 4.* On each resource, a tool keeps an up-to-date status of the computers in the LSDS on which tasks are sent for execution, by constantly interrogating a monitoring system.

*Step 5.* The resources in the cluster run the GA. Each resource starts with a different, specific initialization of the genetic algorithm. The subsequent steps of the GA are similar for all the nodes in the cluster, and so is the fitness formula. The clients will compute different optimum from which the best one will be chosen.

*Step 6.* The migration of the best current solutions is performed after each step of the GA, thus ensuring that the population finds a better optimum. The resources exchange the fittest individuals and insert them into the next generation.

*Step 7.* The reproduction process stops after a finite, predefined number of steps. Each resource in the cluster computes its optimal individual.

*Step 8.* Each resource sends its optimum to all the other nodes in the cluster and the final optimal individual is decided.

*Step 9.* The scheduling obtained is saved in a history file on each resource in the cluster of resources.

The *fitness function* is an essential element of proposed GA. It gives an appreciation of the quality of a potential solution according to the problem's specification. For the scheduling problem, the goal is to obtain task assignments that ensure minimum execution time, maximum processor utilization, a well-balanced load across all machines, and last but not least to ensure that the precedence of the task's is not violated. According to the chromosome encoding and genetic operators presented previously all individuals respect the task dependencies, so the focus should be on the other goals of the problem. The fitness function has the following representations: (1) $F = \sum_i c_i f_i$ or (2) $F = \prod_i f_i$, where $f_i$ encode a criterion in fitness function and $c_i$ is a weight for a criterion ($\sum_i c_i = 1$). In both cases, if $0 \leq f_i \leq 1$, then $0 \leq F \leq 1$. For the proposed genetic scheduling algorithm three criteria are considered: *load balancing* over the resources, $f_1 = t_{\min}/t_{\max} = \min_i\{t_i\}/\max_i\{t_i\}$, *average idle time* of the resources, $f_2 = (1/n)\sum_{i=1}^{n} t_i/t_{\max}$, and the *schedule penalty*, $f_3 = |\text{Sched}|/|T|$, where |Sched| represents the length of a schedule (the number of tasks that respect deadlines and resource restrictions) and $|T|$ is the total number of tasks.

## 5. Experimental Methodology and Results

### 5.1. Simulation Environment

Due to the complexity of the LSDS, involving many resources and many jobs being concurrently executed in heterogeneous environments, there are not many simulation tools to address the general problem of LSDS computing. The simulation instruments tend to narrow the range of simulation scenarios to specific subjects, such as scheduling or data replication. The simulation model provided by MONARC is more generic than others, as demonstrated in [18]. It is able to describe various actual distributed system technologies and provides the mechanisms to describe concurrent network traffic, to evaluate different strategies in data replication, and to analyze job scheduling procedures. In order to provide a realistic simulation, all the components of the system and their interactions were abstracted.

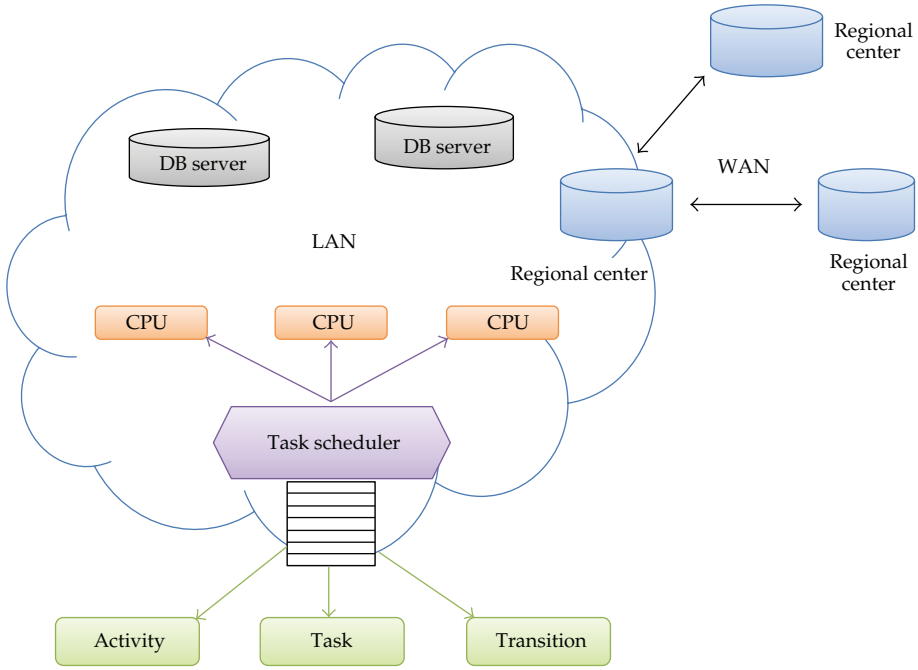


**Figure 2:** MONARC simulation tool: the Regional center model for LSDS control.

The chosen model is equivalent to the simulated system in all the important aspects. A first set of components was created for describing the physical resources of the distributed system under simulation. The largest one is the regional center (see Figure 2), which contains a site of processing nodes (CPU units), database servers and mass storage units, as well as one or more local and wide area networks.

The maturity of the simulation model was demonstrated in previous work. For example, a number of data replications experiments were conducted in [17], presenting important results for the future LHC experiments, which will produce more than 1 PB of data per experiment and year, data that needs to be then processed. In [19] the simulation model was used to conduct a series of simulation experiments to compare a number of different scheduling algorithms.

## *5.2. Evaluation Criteria for LSDS Control*

It is quite difficult to make a comparison among different control systems for LSDS, since each of them is suitable for different situations. For different control systems, the class of targeted applications and LSDS resource configurations may differ significantly. The adequate evaluation criteria for LSDS control systems are as follows. (i) *Application Performance Promotion* involves reviewing how well the applications can benefit from the deployment of the control system (ii) *System Performance Promotion* concerns how well the whole system can benefit (iii) *Control and Efficient Allocation* it is desired so that the LSDS control system can always produce good allocation. However, it is also required that the scheduling system should introduce additional overhead as low as possible. (iv) *Reliability*—a reliable LSDS control system should provide some level of fault tolerance. An LSDS

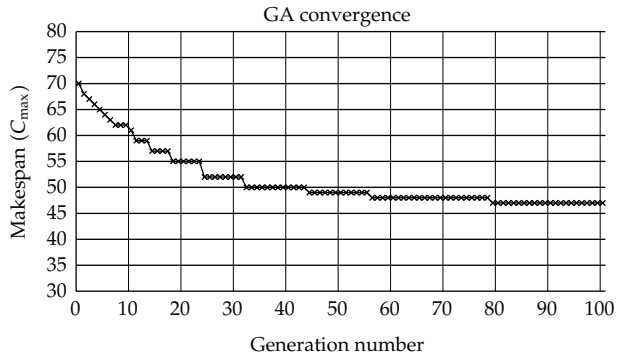


**Figure 3:** Makespan comparison for the scheduling of 38 tasks on 8 processors.

is a large collection of loosely coupled resources, and therefore it is inevitable that some of the resources may fail due to diverse reasons. The control system should handle such frequent resource failures. For example, in case of resource failure, the control system should guarantee an applications completion. (v) *Scalability*—since an LSDS environment is in nature heterogeneous and dynamic, a scalable scheduling infrastructure should maintain good performance with not only increasing number of applications, but also increasing number of participating resources with diverse heterogeneity.

When designing the control infrastructure for LSDSs, these criteria are expected to receive careful consideration. Emphasis may be laid on different concerns among these evaluation criteria according to practical needs in real situations. The performance of scheduling algorithms for LSDS control is usually estimated using a certain number of standard parameters, like total time or schedule length. In the tests performed we used the following evaluation parameters [20, 21]:

(i) total schedule length (SL)—$C_{max}$;

(ii) convergences time—the number of generations needed to obtain performances better then a certain threshold;

(iii) load balancing (where $u_{med}$, denotes the average utilization for all processors in the system):

$$L = 1 - \frac{1}{u_{med}} \sqrt{\frac{1}{n} \sum_{i=1}^{n} (u_i - u_{med})^2}, \quad 0 < L \leq 1. \tag{5.1}$$

The load balancing of system, for a given schedule, converges to 1 when all resources have approximately the same utilization rate, equal to makespan. In these conditions the square deviation $\Delta \to 0$.

### *5.3. Experimental Results*

The test case considered task dependencies containing 38 tasks. The processors' topology contained 8 processors connected in a full mesh. The results presented in Figure 3 show that

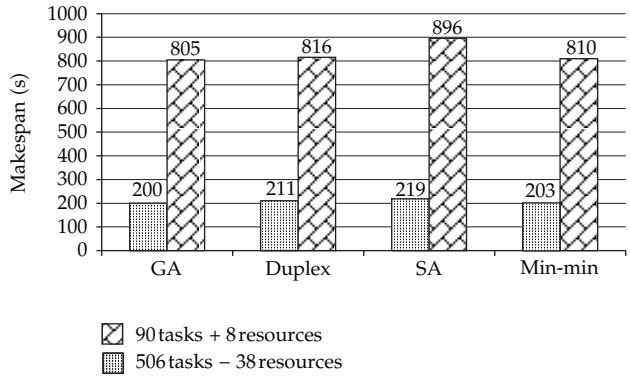


**Figure 4:** Makespan comparison for the following scenarios: 90 tasks + 8 resources, 506 tasks − 38 resources.

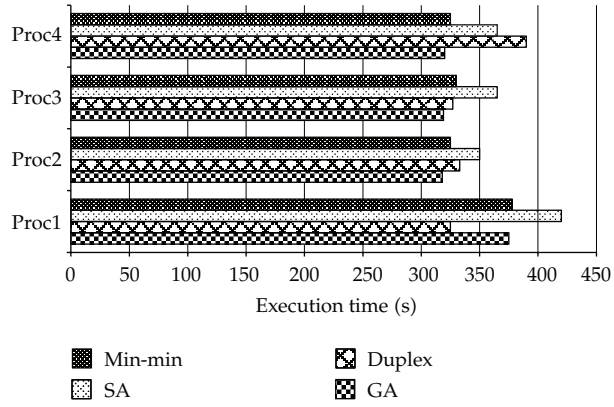


**Figure 5:** Processor utilization overview for 90 tasks.

the genetic algorithm has a very good convergence (after 50 generations there is no significant improvement, so the algorithm could be stopped).

In order to analyze the schedule length of different dependent task scheduling algorithms, it has been used a processor topology containing 8 processors connected in a full mesh. Two tests have been run for DAGs containing 90 and 506 tasks (see Figure 4).

For the first test (90 tasks—left side of the figure), the best result, 263 time units, was provided by the proposed GA. On the second place came the GA without the initialization phase with the value of makespan equal to 266. From the classic scheduling algorithm, Duplex provided the best solution equal to 281, while the result of SA was the worst equal to 292. The test containing 506 tasks was an extreme test. The best result, 1073 time units, was offered by GA, proving once more the importance of the proposed algorithm. The worst solution was given by SA. The other compared algorithm is min-min [22] (see Figure 5).

Memory is another important factor since it is the characteristic that controls most of the allocation algorithms and also since it cannot be oversubscribed. As can be seen, the memory allocator gets to the maximum memory value slower and thus allows for better

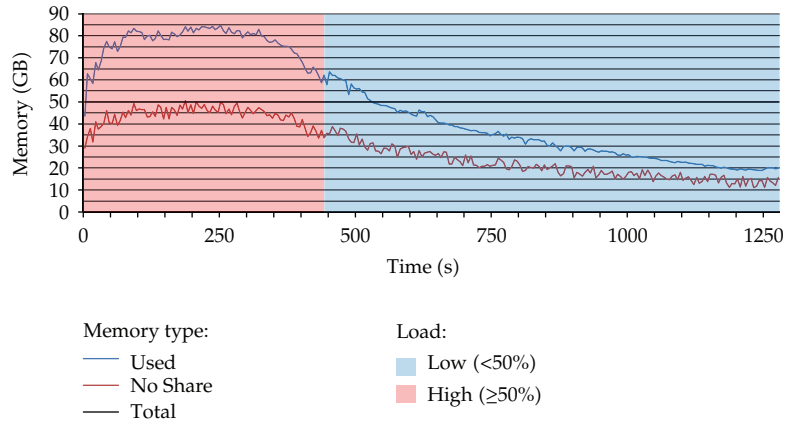


**Figure 6:** Memory usage for Best Fit allocation.

performance. Also this allocator is the first to leave the maximum value barrier when the load is decreased (see Figure 6).

## 6. Conclusions

We present in this paper an algorithm for controling the resources allocation for special tasks type (transitions) in LSDS (considered to be a complex one). The novelty of the proposed algorithm is represented by the multicriteria optimization fitness function for special tasks with specific requirements and constrains. The process was modulated using a genetic scheduling algorithm. The paper analyzed the existing methods for control optimization in LSDS. The multidimensional optimization criteria were considered with the real-time behavior introducing two measures for evaluation: penalty and reward. In accordance with this behavior, the convergence of proposed control method is very good in terms of convergence, solution cost, and memory usage.

The most important contribution of this paper is the innovative method for the optimization of dependent task scheduling control in LSDS. Inspired from the natural models, this algorithm evolves an initial population of chromosomes in order to achieve a good average fitness for the population. The experimental results have proven that the proposed algorithm offers the best solutions in most cases. For comparison were used several classical algorithms such as SA, Duplex, and min-min.

## Acknowledgments

The research presented in this paper is supported by the Romanian Project: *SORMSYS-Resource Management Optimization in Self-Organizing Large-Scale Distributes Systems* (Contract no. 5/28.07.2010, Project CNCSIS-PN-II-RU-PD ID: 201). The work has been cofunded by the Sectorial Operational Program Human Resources Development 2007–2013 of the Romanian Ministry of Labor, Family and Social Protection through the Financial Agreement *POSDRU/89/1.5/S/62557*.

## References


[1] C. Toma, E. G. Bakhoum, and M. Li, "Propagation phenomena and transitions in complex systems: Efficient mathematical models," *Mathematical Problems in Engineering*, vol. 2012, Article ID 429129, 3 pages, 2012.

[2] E. G. Bakhoum and C. Toma, "Specific mathematical aspects of dynamics generated by coherence functions," *Mathematical Problems in Engineering*, vol. 2011, Article ID 436198, 10 pages, 2011.

[3] J. Kołodziej and F. Xhafa, "Integration of task abortion and security requirements in GA-based meta-heuristics for independent batch Grid scheduling," *Computers and Mathematics with Applications*, vol. 63, no. 2, pp. 350–364, 2012.

[4] Y. Huang, N. Bessis, P. Norrington, P. Kuonen, and B. Hirsbrunner, "Exploring decentralized dynamic scheduling for grids and clouds using the community-aware scheduling algorithm," *Future Generation Computer Systems*. In press.

[5] H. Abdu, H. Lutfiyya, and M. A. Bauer, "Optimizing management functions in distributed systems," *Journal of Network and Systems Management*, vol. 10, no. 4, pp. 505–530, 2002.

[6] T. L. Casavant and J. G. Kuhl, "A taxonomy of scheduling in general-purpose distributed computing systems," *IEEE Transactions on Software Engineering*, vol. 14, no. 2, pp. 141–154, 1988.

[7] R. Armstrong, D. Hensgen, and T. Kidd, "The relative performance of various mapping algorithms is independent of sizable variances in run-time predictions," in *Proceedings of the Seventh Hetero-geneous Computing Workshop (HCW '98)*, pp. 79–87, IEEE Computer Society, Washington, DC, USA, 1998.

[8] J. Wu, X. Xu, P. Zhang, and C. Liu, "A novel multi-agent reinforcement learning approach for job scheduling in Grid computing," *Future Generation Computer Systems*, vol. 27, no. 5, pp. 430–439, 2011.

[9] T. Wang, X.-S. Zhou, Q.-R. Liu, Z.-Y. Yang, and Y.-L. Wang, "An adaptive resource scheduling algorithm for computational grid," in *Proceedings of the IEEE Asia-Pacific Conference on Services Computing (APSCC '06)*, pp. 447–450, IEEE Computer Society, Washington, DC, USA, 2006.

[10] B. Sotomayor, R. S. Montero, I. M. Llorente, and I. Foster, "Virtual infrastructure management in private and hybrid clouds," *IEEE Internet Computing*, vol. 13, no. 5, pp. 14–22, 2009.

[11] W. Zhao, K. Ramamritham, and J. A. Stankovic, "Preemptive scheduling under time and resource constraints," *IEEE Transactions on Computers*, vol. 36, no. 8, pp. 949–960, 1987.

[12] J. Koodziej and F. Xhafa, "Enhancing the genetic-based scheduling in computational grids by a structured hierarchical population," *Future Generation Computer Systems*, vol. 27, no. 8, pp. 1035–1046, 2011.

[13] T. Ma, Q. Yan, W. Liu, and C. Mengmeng, "A survey on grid task scheduling," *International Journal of Computer Applications in Technology*, vol. 41, no. 3-4, pp. 303–309, 2011.

[14] G. Logothetis, *Specification, Modelling, Verification and Runtime Analysis of Real Time Systems*, IOS Press, 2004.

[15] E. Fersman and W. Yi, "A generic approach to schedulability analysis of real-time tasks," *Nordic Journal of Computing*, vol. 11, no. 2, pp. 129–147, 2004.

[16] G. C. Buttazzo, *Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications*, Springer, 3rd edition, 2011.

[17] G. V. Iordache, M. S. Boboila, F. Pop, C. Stratan, and V. Cristea, "A decentralized strategy for genetic scheduling in heterogeneous environments," *Multiagent and Grid Systems*, vol. 3, no. 4, pp. 355–367, 2007.

[18] C. Dobre, C. Stratan, and V. Cristea, "Realistic simulation of large scale distributed systems using monitoring," in *Proceedings of the 7th International Symposium on Parallel and Distributed Computing (ISPDC '08)*, pp. 434–438, IEEE Computer Society, Washington, DC, USA, July 2008.

[19] F. Pop, C. Dobre, G. Godza, and V. Cristea, "A simulation model for grid scheduling analysis and optimization," in *Proceedings of the International Symposium on Parallel Computing in Electrical Engineering (PARELEC '06)*, pp. 133–138, IEEE Computer Society, Washington, DC, USA, 2006.

[20] S. Venugopal and R. Buyya, "An SCP-based heuristic approach for scheduling distributed data-intensive applications on global grids," *Journal of Parallel and Distributed Computing*, vol. 68, no. 4, pp. 471–487, 2008.

[21] F. Pop, C. Dobre, and V. Cristea, "Performance analysis of grid DAG scheduling algorithms using MONARC simulation tool," in *Proceedings of the 7th International Symposium on Parallel and Distributed Computing (ISPDC '08)*, pp. 131–138, IEEE Computer Society, Washington, DC, USA, July 2008.

[22] F. Xhafa and A. Abraham, "Computational models and heuristic methods for Grid scheduling problems," *Future Generation Computer Systems*, vol. 26, no. 4, pp. 608–621, 2010.