Logic Journal of IGPL Advance Access published February 4, 2011

Dynamic model of distribution and organization of activities in multi-agent systems

CAROLINA ZATO, ANA DE LUIS, JAVIER BAJO, JUAN F. DE PAZ and JUAN M. CORCHADO, *Department of Computer Science and Automation, University of Salamanca, Plaza de la Merced, s/n, 37008, Salamanca, Spain. E-mail: carol_zato@usal.es; adeluis@usal.es; jbajope@usal.es; fcofds@usal.es; corchado@usal.es*

Abstract

Currently, the allocation of tasks is a problem in many different areas such as e-Goverment. Traditionally, the assignment is done manually; therefore it is necessary to anticipate workloads and employee characteristics. This article describes a system based on virtual organizations of agents that allows recommendations about planning of tasks to minimize the resources necessary for their accomplishment and to obtain the maximum profit. For this purpose, a hybrid artificial intelligence system with genetic algorithm, queuing theory and case-based reasoning is used to obtain an efficient distribution. The final part of the article is focused on validating the plan developed inside a case of study centered in the e-Government in order to obtain empirical results.

Keywords: Multi-agent systems, virtual organizations, queuing theory, genetic algorithm, scheduling, e-Government.

1 Introduction

Planning and distribution of tasks is a problem that can be encountered in various activities [19, 22]. These works seek to optimize the allocation of resources to maximize efficiency and minimize costs. The main problem arises from the dynamism of the working scenarios and the difficulty of finding a balance between time spent on planning and on implementing plans, which particularly in these environments, is the key to adapt to the needs of the organization requiring to replan frequently. Therefore, it is necessary to create a system that predicts demand for resources so it can perform tasks allocation maximizing benefits and minimizing delays.

Traditionally, the techniques used for planning and resource allocation are performed using exact methods. The exact methods such as linear programming [7], non-linear programming [16, 20] and graph theory, ensure optimal solutions in execution time depending on the number of existing variables. Thus, such techniques, as linear programming or non-linear, are not applicable to problems of the nondeterministic polynomial time (NP)-hard type. Moreover, these techniques have difficulties defining constraints and objective functions, as the objective function must present all the existing combinations to define the variables and this number of combinations can be very high. It is therefore advisable for such types of problems to use metaheuristic techniques [8, 10, 18] that allow obtaining efficient solutions in reasonable execution times. Hybrid artificial intelligence systems can incorporate metaheuristics and artificial intelligence techniques, and they are used in a great variety of problems [4, 5, 11].

doi:10.1093/jigpal/jzr005

This article proposes a hybrid artificial intelligence system able to estimate work demands in order to estimate the number of resources needed and according to these claims carry out a work-resource distribution to maximize benefits and minimize delays. The hybrid artificial intelligence system is performed by applying case-based reasoning (CBR) systems [1, 9]. The CBR system integrates into the various stages of reasoning artificial intelligence techniques to estimate resources based on queuing theory and planning using genetic algorithms. This planning mechanism is applied to virtual organizations (VOs) [6] of agents to simulate the behaviour of organizations in planning and allocation of work a e-Government scenario when, given the large number of procedures and users of this entity, it is essential to have a good planning system that can do the work successfully and on time, devoting staff just enough to meet the challenges ahead.

This article is divided as follows: Section 2 describes multi-agent systems (MASs) and planning mechanisms used for assigning dynamic tasks, Section 3 presents the proposed model and Sections 4 and 5 describe the results obtained and the conclusion.

2 Multi-agent systems

A MAS is basically a network of organizations focused on solving problems and working together to find answers to problems that are beyond the individual capabilities or knowledge of each entity. In our case, these entities are case-based reasoning-beliefs, desires, intentions (CBR-BDI) agent, which gets its name from the BDI architecture with a CBR reasoning system [3, 9]. A CBR manages cases (past experiences) to solve new problems. The way cases are managed is known as the CBR cycle, and consists of four sequential stages that are recalled every time a problem needs to be solved: retrieve, reuse, revise and retain. Each of the steps of the CBR life cycle requires a model or method in order to perform its mission.

An open MAS [2] should allow the participation of heterogeneous agents, which change over time, with architectures and even with different languages. For this reason, we cannot rely on agents' behaviour when it is necessary to establish controls on the basis of norms or social rules. For this and because of the characteristics of open environments, new approaches are needed to support evolutive systems, and to facilitate their growth and runtime updates especially due to the dynamics of open environments. This is one of the reasons that encourage the use of VOs. A VO [6] is an open system designed for grouping, for the collaboration of heterogeneous entities and where there is a separation between form and function that defines their behaviour. The concept of organization is seen as a promising solution to manage the coordination of the agents and control their behaviours and actions.

Every organization needs coordination support to determine explicitly how to organize and carry out the actions and tasks within it. In this work, due to the use of THOMAS [2], organizations are comprised of organizational units of an agent of hierarchy type where the supervisor has control over other members, coordinating the tasks, centralizing planning and decision-making.

2.1 Planning tasks

Planning problems are usually performed by exact techniques such as linear programming, which can be used as in the Simplex algorithm [7], quadratic programming or by using other non-linear programming techniques such as Lagrange multipliers [20], Kuhn–Tucker [16] or allocation problems in graph theory and application of algorithms such as Ford–Fulkerson [17]. These problems are included within the area known as combinatorial optimization and in most cases these problems belong to the NP-hard family problems. This situation justifies the application of heuristic algorithms for the search of solution in reasonable time.

Heuristic algorithms are easily generalized to several types of problems which are called metaheuristics. Among the algorithms is the GRASP algorithm, tabu search, simulated annealing or genetic algorithms. Greedy algorithms (AVM), such as GRASP, iteratively generate solutions from a randomly generated greedy solution, which is modified to optimize it [8] and tabu search [18] iteratively adjust a particular solution from the local search of optimal solutions; these algorithms have the problem of local minima and for this we have the annealing algorithms [10] to explore alternative solutions to local optimum in a probabilistic way in order to avoid local minima. Genetic algorithms [21] include characteristics of the previous metaheuristics and initially generate multiple random solutions which are iteratively altered by mutation and crossover operators and explore solutions locally by mutations to avoid falling into local minima.

In recent years, we have proposed a number of approaches [15] to model and solve the various problems of scheduling tasks, with varying degrees of success. Reviewing various comparisons [14] of the efficiency of different metaheuristics methods, it is possible to see that any single case, a method is clearly distinct from another.

3 Proposed model

The model proposed in this article focuses on developing a planning mechanism to coordinate the agents found in the VO. Thus, first the roles that these agents can take are set out:

Processor role: responsible for carrying out the activities required for each specific task. For this reason, the agent will specialize depending on the type of tasks the system must solve.

Planner role: it designs the overall plan to be implemented by the organization. It sets the number of processor agents and makes the distribution of tasks depending on the role they play. It also replans depending on the size of the input queue or inability to accomplish with a plan.

Distributor role: its aim is the distribution of tasks according to its completion by the agents and checking that each task is being processed within time limits to serve the plan.

Coordinator: it performs the general control of the system. It communicates with the THOMAS platform elements to carry out control actions (connect, disconnect, exceptions, etc.).

Manager role: This agent manages all the information of the task and communicates to the user.

In the Figure 1, the different agents of the system and the interactions among them are represented. In the upper corner of the figure, we can see the tasks list that store the activities to carry out in the MAS and in the centre of the image, the agents and the interconnections are showed.

Each task in the list contains the following information: TaskID, TaskType activity identifier, AccruedBenefit monetary amount of work done previously would be lost if we do not meet deadlines, date of entry, maximum resolution Term, Duration time used in performing the task.

To get over the planning process the agent follows the CBR-BDI planning model. The first point in the definition of a CBR-BDI model is the definition of case (1).

$$C=\{t_i/t_i=(ids,idt,b,B,f,p,d,a), i=1...n\} \tag{1}$$

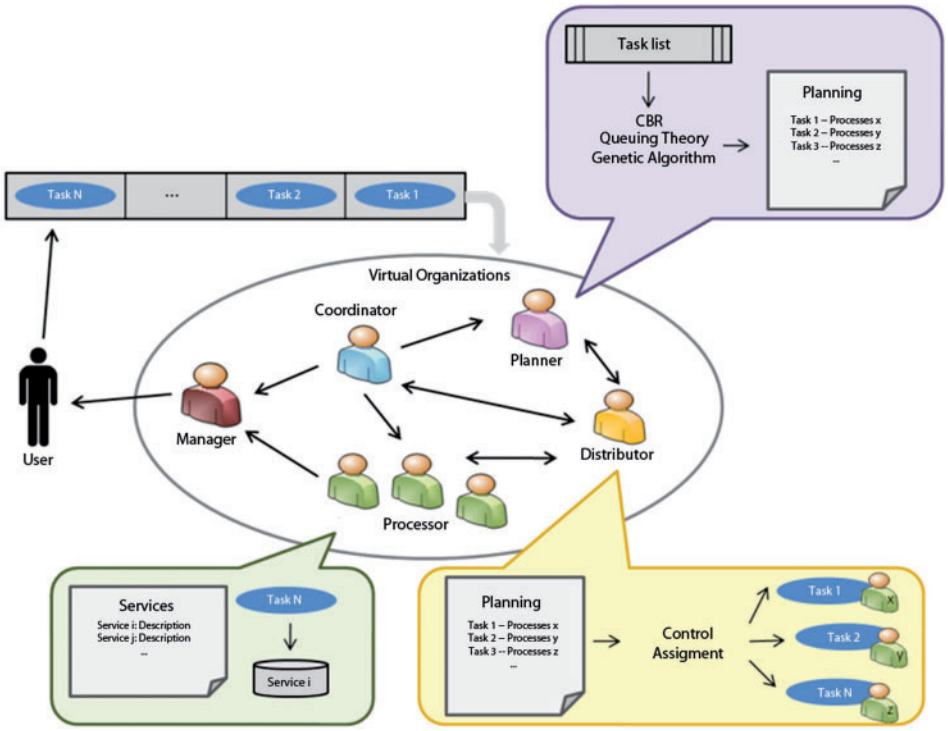

FIG. 1. Outline of proposed model.

where t_i represents the task i, *ids* is the identifier of the task type, *idt* the process, b benefit, B the accrued benefit , f entry, p term, d duration and a the ID of the agent that performed the task. The application of the different stages of reasoning is performed as follows:

Recovery: during this stage we proceed to retrieve the most similar cases to the current case c_{n+1}. The most similar cases are those containing more tasks of the same type as the tasks which are currently queued in the system. The number of cases retrieved is predefined to subsequently implement the adaptation phase. The set of recovered cases is called $C_r \subseteq C$.

Adaptation: during this stage, the recovered information C_r is adapted from cases of the memory of cases to generate the entire plan corresponding to the case c_{n+1}. The information recovered is adapted by applying queuing theory and genetic algorithms. On the one hand, the recovered information is used to determine the arrival rate and service for each of the tasks and, thus, using queuing theory it is possible to determine the number of agents needed to run the indicated tasks. Recovered cases are the basis for constructing initial chromosomes in genetic algorithms.

Revise: the revise stage is automatically performed as tasks are being finalized by the agents. The agent updates the duration of the tasks as they are completed and, in turn, makes new plans if any notice is received from the processor agents on the inability to complete a plan under time constraints provided.

Retain: the retain phase is limited to store the case when the day is finished. The new case memory C' is defined as follows: $C' = C \cup c_{n+1}$ To limit the size of the memory of cases, some of them are removed from memory if exceeded a predefined age.

3.1 Dynamic planning roles

The number of agents that should be available in the system is dynamically estimated. It is intended that the number of agents suits demand to ensure that the system utilization factor ρ is less than 1. This estimate will be done through the use of queuing theory in a M/G/s model, where the arrival rate follows a Poisson distribution (the most commonly used in similar work [13]), the exponential service and the existence of multiple servers (agents).

The problem of planning multiple tasks can be reduced to the case of planning for a single task of each type. Thus, for each task a planning is performed independently to independently calculate the average waiting time and the average queue length. The average waiting time and the overall average length is reduced for calculating the average values calculated for each of the tasks. In the case of the M/G/s model, where $s=1,2,3,\ldots$ is the number of agents and given an arrival rate $\lambda_n=\lambda=\text{cte}$, the service rate when there are n processes is defined by the following (2) [12].

$$\mu_n=\begin{cases} n\mu & n=1,2,\ldots,s-1 \\ s\mu & n\geq s \end{cases} \tag{2}$$

where μ represents the average service rate for s available agents. This value depends on both the agents and the machine found.

Assuming that the system is in a stable condition, i.e. it meets the utilization factor $\rho=(\lambda/\mu s)<1$, the probability that n tasks exists (P_n) in the system is given by (3) [12].

$$P_n=c_nP_0=\begin{cases} \frac{\lambda^n}{n!\mu^n}P_0 & n=0,1,\ldots,s-1 \\ \frac{\lambda^s}{s!\mu^s}\left(\frac{\lambda}{s\mu}\right)^{n-s}P_0 & n\geq s \end{cases} \tag{3}$$

where:

$$P_0=\frac{1}{\sum_{n=0}^{s-1}\frac{\lambda^n}{n!}+\frac{\lambda^s}{s!\mu}\frac{1}{1-\frac{\lambda}{s\mu}}} \qquad c_n=\begin{cases} \frac{\lambda^n}{n!\mu^n} & n=1,2,\ldots,s-1 \\ \frac{\lambda^s}{n!\mu^s}\left(\frac{\lambda}{s\mu}\right)^{n-s} & n\geq s \end{cases} \tag{4}$$

Having defined the probability that n tasks exist in the system, it is possible to define the number of tasks L_q in the system queue and average waiting time W_q of tasks in the tail of the system (5) [12].

$$L_q=\sum_{n=s}^{\infty}(n-s)P_n=P_0\frac{\lambda^s}{s!\mu^s}\frac{\lambda}{s\mu}\frac{1}{\left(1-\frac{\lambda}{s\mu}\right)^2}W_q=\frac{L_q}{\lambda} \tag{5}$$

To determine the optimal number of agents, an estimate that minimizes the cost function is made, that depends on the number of agents used and on the waiting time in the queue. The function is defined in a particular way for each service, depending on the actual costs of each agent in the system, though the following benefit function is provided (6).

$$f(L,P_0,\ldots,P_{s-1},\mu',\bar{p},\bar{b})=f_b(L,\mu',\bar{p},\bar{b})-k\cdot s \tag{6}$$

$$f_b(L,\mu',\bar{p},\bar{b})=\begin{cases} (\bar{p}/u')\bar{b}\cdot s\cdot(1-\rho) & \text{si L}\cdot u'>\bar{p}\cdot s\cdot(1-\rho) \\ L\bar{b} & \text{si L}\cdot u'\leq\bar{p}\cdot s\cdot(1-\rho) \end{cases} \tag{7}$$

where k is a constant associated with the cost of having an agent working, $\bar{b}$ the average benefit of performing the task, μ' is the average time to complete the task obtained from the service rate and $\bar{p}$ the average time to execute a task. If the conditions of stability are overpassed, f_b is counted only up to the utilization factor 1. The utilization factor ρ varies according to the new services added to the queue till it reaches the utilization factor of 1.

Following the cost function given in (6), the global cost function (8) is introduced, which takes into account the implementation of the various services.

$$f(f_1,\ldots,f_k)=\sum_{j=1}^{k} f_i \tag{8}$$

where f_i is calculated from (7). Because sometimes you might not be given the stability conditions, it is necessary to calculate the terms in order of benefit depending the type of the task so that when the utilization factor of 100% is reached, it is possible to calculate the summation terms. The optimization function is defined in (9). The maximum value is calculated iteratively starting with number of agents equal to 1, and the fixed value is the first local maximum that corresponds to the global maximum.

3.2 Task assignment

Once the number of starting agents is considered to minimize costs, it is possible to proceed to make an allocation of tasks between the available agents. If the system utilization factor does not exceed the value of 1, the distribution of tasks among agents is performed so as to ensure as far as possible that it can perform assigned tasks in case of delays or the time to perform a task increases. It is performed so as to maximize the following function (9):

$$\max\sum_{i=1}^{k} f_i \quad \text{where} \quad fi=\begin{cases}\log(1-|x_i-\bar{x}|) & (x_i-\bar{x})\geq 0\\ -\log(1-|x_i-\bar{x}|) & (x_i-\bar{x})<0\end{cases} \tag{9}$$

where $x_i=t_i-a_{i-1}-c_i$ with t_i the maximum time for completion of the task i, a_{i-1} the cumulative time to perform the tasks $i-1$ above and finally c_i the time to run the task i, which is customized according to the agent selected and calculated from the average value of previously executed tasks. Minimizing the differences, get all the tasks to have a uniform distribution of the remaining time so it gets easier to achieve them.

If the system utilization factor is up to 1, the aptitude function is redefined to minimize possible losses of the work already done

$$\min\sum_{i=1}^{k} -B_i \tag{10}$$

As in the previous case, to know the value of each B_i it is necessary to establish the execution order of the procedures. If there is enough time for completing the task i, the value of B_i is ignored.

The chromosome encoding is performed so that each gene is composed of the elements listed by t_i identified in (1). The crossover operator is defined similarly to the multi-point used in problems such as travelling salesman problem (TSP).

Mutation operators define various modes that will be executed randomly, and just those mutations that improve the aptitude of chromosomes will be selected. Definite mutation operators are to: exchange order of tasks, exchange of assigning contiguous tasks and changing the allocation of a task.

Elitism operator is defined to keep the percentage of efficient solutions in every generation of population. The roulette selection is the criteria chosen for this.

4 Case study and results

The case study presents a society oriented system to process all cases that reach the public administration by electronic means. The implemented system is responsible for conducting a simulation of the behaviour of the Planner agent within the VO. In order to assess the mechanism of the proposed planning, taking into account that the final goal is to process all the records within time limits or, if not possible, to maximize the benefit being carried out by different tests. The system was tested with two different records for four simulation modes: Mode 1—without planning, Mode 2—calculating the number of agents needed within queuing theory, Mode 3—planning (including queuing theory and genetic algorithms) and Mode 4—the whole planning with CBR. In the first case (Test 1), a list of 500 records was entered within a period of 120 min and the second case (Test 2) had 1500 records into the system within 210 min. Previously, it had a memory of 700 cases, based on cases where values were altered randomly following a normal distribution.

Figure 2a shows the comparative number of cases that could not be processed in time for each mode in the two tests. It can be seen that the qualitative leap occurs in both cases, by the introduction in the third mode of simulation of genetic algorithms to carry out the distribution. The introduction of the CBR also produces an improvement in the last mode so as to work with parameters that fit closer to reality due to learning by the system. Test 2 for Mode 4 shows that the number of unprocessed cases on time is higher than Test 1, this is mainly because the recovered arrival rate exceeded the records retrieved from the memory of the CBR.

Figure 2b is even more significant because it reveals the benefits obtained in different modes. To assign a value to the benefit field for each record a point system is used taking into account various aspects such as the type of record, the economic gain expected, previous work, etc.

As expected, the worst benefit is obtained in a without planning mode. In this case, the fact that a generic estimate of the number of players is very vague so in many cases there are losses by not having enough agents or having too many. Also since it does not take into account the benefit of each records, it gives priority to some records which are considered losses. In the second case, the use of queuing theory improves the benefits being able to balance between the number of agents required and ordination benefit, and thus assigning priority to improve the figures. In the third mode,

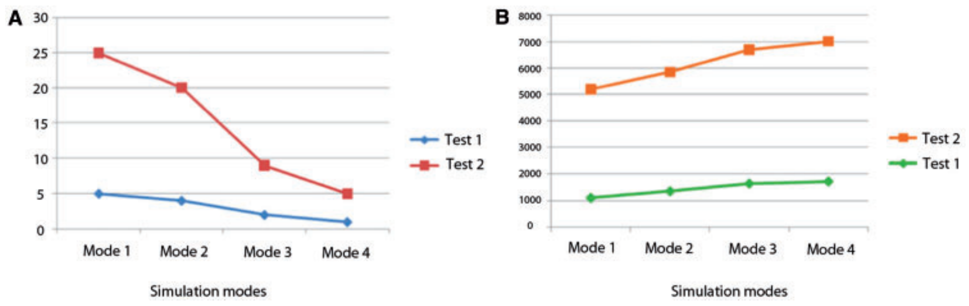

FIG. 2. (**A**) Number of files out of time. (**B**) Profit earned.

as the above, chart shows that the introduction of a task distribution based on genetic algorithm is the strong point of planning. Finally, still better benefits are obtained taking into account the adaptation from past cases provided by the CBR.

5 Conclusions and future work

Throughout this article, we introduced a model of VO with a mechanism for planning tasks for reference when distributing tasks in a dynamic environment. Using a genetic algorithm as a mechanism for the exploration of search domain and for optimization has been effective within the proposed problem characterized by its complexity, given the large number of available combinations. Using queuing theory allowed setting the number of agents required for resource optimization. Both methods have been successfully incorporated into the reasoning cycle of CBR, specifically, in the adaptation phase. The results confirm that planning leads to an improvement and by the refinement of the cases included in the reasoning cycle of CBR, it will be possible to achieve a high level of learning and adaptation in a dynamic environment. In the short term, the proposed future work is to explore other heuristics and metaheuristics methods in order to try different formulas for the allocation of tasks, setting an efficiency value to each method and recommending to the user different planning methods associated with its success rate.

Funding

This work has been supported by the MICINN TIN 2009-13839-C03-03.

References

[1] Abraham, E. Corchado, and J. M. Corchado. Hybrid learning machines. *Neurocomputing*, **72,** 2729–2730, 2009.

[2] J. Bajo, J. J. Corchado, V. Botti, and S. Ossowski. Practical applications of agents and MAS: methods, techniques and tools for open MAS. *Journal of Physical Agents*, **3**, 1–2, 2009.

[3] E. Corchado, M. Assumpcio, and M. L. Borrajo. A maximum likelihood Hebbian learning-based method to an agent-based architecture. *International Journal of Computer Mathematics*, **86,** 1760–1768, 2009.

[4] E. Corchado, A. Abraham, and A. Carvalho. Hybrid intelligent algorithms and applications. *Information Science*, **180,** 2633–2634, 2010.

[5] E. Corchado, A. Arroyo, and V. Tricio. Soft computing models to identify typical meteorological days. *Logic Journal of the IGPL*, 2010 [Epub ahead of print, doi:10.1093/jigpal/jzq035].

[6] M. Esteva, J. Rodr´ıguez, C. Sierra, P. Garcia, and J. Arcos. On the formal specific ations of electronic institutions. In *Agent-mediated Electronic commerce. Lecture Notes in Computer Science*, pp. 126–147. Springer, 2001.

[7] S. Kabadi and A. Punnen. A strongly polynomial simplex method for the linear fractional assignment problem. *Operations Research Letters*, **36,** 402–407, 2008.

[8] T. Kacprzak, K. Walkowiak, and M. Wozniak. GRASP algorithm for optimization of grids for multiple classifier system. *Advances in Soft Computing*, **73**, 137–144, 2010.

[9] J. Kolodner. *Case-Based Reasoning*. Morgan Kaufmann, 1993.

[10] M. Kolonko. Some new results on simulated annealing applied to the job shop scheduling problem. *European Journal of Operational Research*, **113,** 123–136, 2009.

[11] J. Legarreta, F. Boto, I. Macía, J. Maiora, G. García, C. Paloc, M. Graña, and M. de Blas. Hybrid decision support system for endovascular aortic aneurysm repair follow-up. In *Hybrid Artificial Intelligence Systems*, Vol. 6076 of *Lecture Notes in Computer Sciences*, pp. 500–507. Springer, 2010.
[12] Q. Martín. *Investigación Operativa*. Prentice-Hall, 2003.
[13] D. A. Menasce. Trade-offs in designing web clusters. *Internet Computing, IEEE*, **6**, 76–80, 2002.
[14] P. Rodríguez. Discusión y Análisis de la metaheurística SN. Depto. Investigación Operativa, InCo, FI, UdelaR. *Reporte Tecnico 03-02*, 2003 [Discussion and Analysis of the SN Metaheuristic. Department of Operations Research, InCo, FI, UdelaR. *Technical Report 03-02*, 2003].
[15] M. Seda. Mathematical models of flow shop and job scheduling problems. In *Worl Academy of Science, Engineering and Technology*, **31,** 122–127, 2007.
[16] C. Shi, J. Lu, and G. Zhang. An extended Kuhn-Tucker approach for linear bilevel programming. *Applied Mathematics and Computation*, **162,** 51–63, 2005.
[17] K. Shin and S. Corder. Implementing the Ford-Fulkerson labeling algorithm with fixed-order scanning. *Computers & Operations Research*, **19,** 783–787, 1992.
[18] E. Taillard. Parallel taboo search technique for the job shop scheduling problem. *Journal on Computing Science*, **6**, 108–117, 1994.
[19] R. R. Trippi, A. W. Ash, and J. V. Ravenis. A mathematical approach to large scale personnel assignment. *Computers & Operations Research*, **1**, 111–117, 1974.
[20] S. Wan, F. Yang, and E. Izquierdo. Lagrange multiplier selection in wavelet-based scalable video coding for quality scalability. *Signal Processing: Image Communication*, **24**, 730–739, 2009.
[21] J. Yu and R. Buyya. Scheduling scientific workflow applications with deadline and budget constraints using genetic algorithms. *Scientific Programming*, **14**, 217–230, 2006.
[22] G. Zülch, S. Rottinger, and T. A. Vollstedt. Simulation approach for planning and reassigning of personnel in manufacturing. *International Journal of Production Economics*, **90,** 265–277, 2004.

Received 25 October 2010