Dynamic Assignation of Roles and Tasks in Virtual Organizations of Agents

Carolina Zato, Ana de Luis, Juan F. De Paz, and Vivian F. López

Abstract. Nowadays, a common problem that affects the workflow and the results of an entity is the planning and distribution of tasks. Doing this manually implies anticipate workloads and employee characteristics, which is inefficient and almost uncalculated in high dynamic environments. In this paper, a model that generates a planning of tasks, minimizing the resources necessary for its accomplishment and obtains the maximum benefits is presented. Within this proposal, genetic algorithms, queuing theory, and CBR are used in different stages to obtain an efficient distribution. To test the system, the chosen case study that fits the scenario, is the e-Government where an elevated number of tasks must be solved in a precise term using the minimal resources.

Keywords: multiagent 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 [1] [2]. The goal of these works is optimize the quantity of resources needed, minimize costs and moreover, get the maximum benefits possible. In this aspect, several variables has to be considered such as the high dynamism of the scenario that forces to find a balance between time spent on frequently planning and on implementing plans to adapt the organization. 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 like linear programming [21], nonlinear [13] [15] programming and graph theory, ensure that you get optimal solutions in execution times depending on the number of existing variables. Thus,

Carolina Zato · Ana de Luis · Juan F. De Paz · Vivian F. López
Department of Computer Science and Automation, University of Salamanca
Plaza de la Merced, s/n, 37008, Salamanca, Spain
e-mail:{carol_zato,adeluis,fcofds,vivian}@usal.es

A. Abraham et al. (Eds.): International Symposium on DCAI, AISC 91, pp. 59–68.
springerlink.com

such are not suitable to problems of the NP-Hard type and have difficulties defining constraints and objective functions. It is therefore advisable for such types of problems to use metaheuristics [17] [18] [19] [20] techniques that allow obtaining efficient solutions in reasonable execution times.

This paper proposes a system able to estimate work demands in order to estimate the number of resources needed and according to these claims carry out a work-resources distribution to maximize benefits and minimize delays. The planning model is performed by applying CBR systems [6]. The CBR system integrates into the various stages of reasoning techniques to estimate resources based on queuing theory and planning using genetic algorithms. This planning mechanism is applied to virtual organizations [5] of agents [3] to simulate the behaviour of organizations in planning and allocation of work in the case study selected e-Government, when given the large number of procedures and users of this entity it is essential to have a good system of planning 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 two describes multi-agent systems and planning mechanisms used for assigning dynamic tasks, section three presents the proposed model, sections four and five describe the results obtained and the conclusion

2 Multi-agent Systems

A multi-agent system (MAS: Multi-Agent System) 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 [3]. In our case, these entities are CBR-BDI agents, which get its name from the BDI architecture with a CBR reasoning system [6] [7]. 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 which 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.

The open MAS [22] 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 the evolution of systems, and to facilitate their growth and run-time updates especially due to the dynamics of open environments. This is one of the reasons that encourage the use of virtual organizations (VO). A VO [5] 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.

3 Planning Tasks

Planning problems are usually performed by exact techniques such as linear programming which can be used as in the Simplex algorithm [21], quadratic programming or by using other nonlinear programming techniques such as Lagrange multipliers [13], Kuhn-Tucker [15] or allocation problems in graph theory and application of algorithms such as Ford-Fulkerson [4]. The nature of this type of problems is combinatorial and therefore the time required to find the optimal solution grows exponentially with the number of tasks considered. 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 a reasonable time.

Heuristic algorithms, which are easily generalized to several types of problems, are called metaheuristics. Metaheuristics are smart strategies to design, improve and optimize very general heuristics, giving high performance with respect to traditional heuristics. Among these algorithms are the GRASP algorithm, taboo search, simulated annealing or genetic algorithms. Greedy algorithms (AVM) like GRASP generate solutions iteratively from a randomly generated greedy solution, which is modified to optimize it [17] [16]. Taboo search [18] [19] adjust iteratively a particular solution from the local search of optimal solutions. These algorithms have the problem of local minima and due to this, the simulated annealing algorithms [20] [18] explore alternative solutions to local optimum in a probabilistic way in order to avoid local minima. Genetic algorithms [10] 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, a number of approaches [12] have been proposed to model and solve several problems of scheduling tasks, with varying degrees of success. Reviewing various comparisons [11] [14] of the efficiency of different metaheuristics methods shows that any single case a method is clearly distinct from another.

4 Proposed Model

The model proposed in this paper focuses on developing a planning mechanism to coordinate the agents included in the VO. The roles of the agents are:

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

2. **Planner role.** Design the overall plan to be implemented by the organization. Sets the number of processor agents and makes the distribution of tasks depending on the role they play. Replan depending on the size of the input queue or inability to accomplish with a plan.
3. **Distributor role.** Distributing tasks according to its completion by the agents and checks that each task is being processed within time limits to serve the plan.
4. **Coordinator.** Performs the general control of the system. Communicate with the platform elements to carry out control actions (connect, disconnect, exceptions, etc.).
5. **Manager role.** This agent manages all the information of the task and communicates to the user.

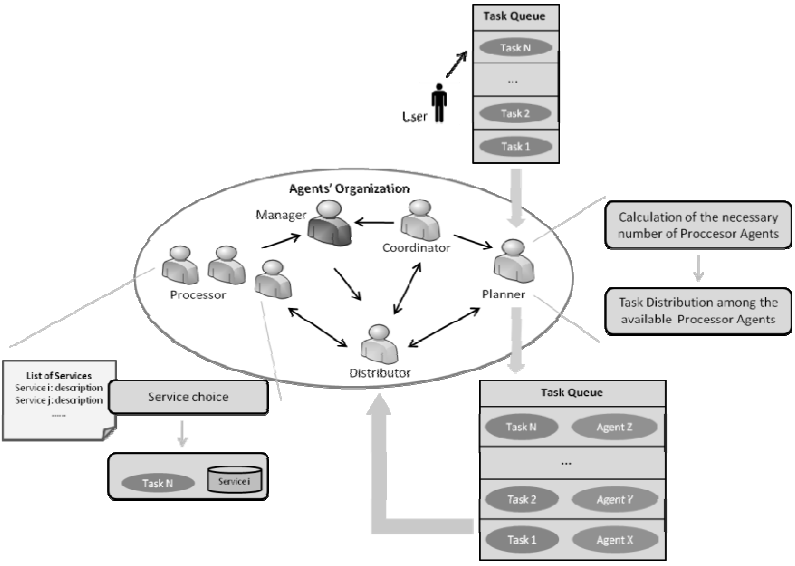

Fig. 1 Outline of proposed model

In the figure 1 the different agents of the system and the interactions among them are represented. In the upper corner of the figure shows the task list that store the activities to carry out in the multi-agent system and in the centre of the image, the agents and the interconnections are showed.

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

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

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 date of entry, p deadline date, d duration, 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 the most similar cases to the current case cn+1 are retrieved. 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 are called $C_r \subseteq C$.

Adaptation. During this stage, the recovered information Cr is adapted from cases of memory to generate the entire plan corresponding to the case cn+1. The information recovered is adapted by applying queuing theory and genetic algorithms. On one hand, the recovered information is used to determine the arrival rate and service for each of the tasks and thus using queuing theory we can determine the number of agents needed to run the indicated tasks. Recovered cases are the basis for constructing initial chromosomes in genetic algorithms.

Review. The review stage is performed automatically as the agents are finalizing tasks. 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.

Learning. The learning 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 cases of memory, some are removed from memory if exceeded a predefined age.

4.1 Dynamic Planning Roles

The number of agents that should be available in the system is estimated dynamically. 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 model M/G/s, where the arrival rate follows a Poisson distribution (the most commonly used in similar work [9]), 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 so as to calculate the average waiting time and average queue length independently. The average waiting time and the overall average length is reduced to calculating the average values calculated for each of the tasks. In the case of the M/G/s model where s = 1,2,3, ... is the number of agents and given an arrival rate $\lambda_n = \lambda$ = cte, the service rate when there are n processes is defined by the following equation (2) [8].

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

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 queuing theory [8] allows to calculate the probability that n tasks exists (Pn) in the system, the number of tasks (L) in the system.

To determine the optimal number of agents, an estimation that minimizes the cost function is calculated and it 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 (4).

$$f(L, P_0,.., P_{s-1}, \mu', \overline{p}, \overline{b}) = f_b(L, \mu', \overline{p}, \overline{b}) - k \cdot s \tag{3}$$

$$f_b(L, \mu', \overline{p}, \overline{b}) = \begin{cases} (\overline{p} / u')\overline{b} \cdot s \cdot (1-\rho) & \text{si L} \cdot u' > \overline{p} \cdot s \cdot (1-\rho) \\ L\overline{b} & \text{si L} \cdot u' \leq \overline{p} \cdot s \cdot (1-\rho) \end{cases} \tag{4}$$

Where k is a constant associated with the cost of having an agent working, $\overline{b}$ the average benefit of performing the task, μ' is the average time to complete the task obtained from the service rate, $\overline{p}$ the average time to execute a task. If the conditions of stability are overpass, fb 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 (4), a global cost function is introduced (5) that takes into account the implementation of the various services.

$$f(f_1,..., f_k) = \sum_{j=1}^{k} f_i \tag{5}$$

Where f_i is calculated from equation (4). Because sometimes you might not be given the stability conditions, it is necessary to calculate the terms in order of benefit depending on the type of the task so that when you reach the utilization factor of 100%, the process ends calculating the summation terms. Once the optimization function defined in (5) the maximum value is calculated iteratively starting with number of agents equal to 1, the fixed value is the first local maximum that corresponds to the global maximum.

4.2 Task Assignment

Once the number of starting agents is considered to minimize costs, an allocation of tasks between the available agents has be done. 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 (6):

$$\max \sum_{i=1}^{k} f_i \; where \; fi = \begin{cases} \log(1-|x_i - \bar{x}|) & (x_i - \bar{x}) \geq 0 \\ -\log(1-|x_i - \bar{x}|) & (x_i - \bar{x}) < 0 \end{cases} \quad (6)$$

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 greater than 1, the aptitude function is redefined to minimize possible losses of the work already done.

$$\min \sum_{i=1}^{k} -B_i \quad (7)$$

As in the previous case to know the value of each Bi it is necessary to establish the order of execution of the procedures. If at the time of completing this task the value of Bi was not taken into account.

The chromosome encoding is performed so that each gene is composed of the elements listed by ti identified in (1). The crossover operator is defined similarly to the multi-junction used in other problems such as TSP. The operator is defined as follows:

- Select a partial route
- Exchange of direct segments of tasks where ID matches
- The exchanges define a series of matches that relates each of the genes of a chromosome with which occupies the same position in the other parent

Mutation operators define various modes that will be executed randomly, and just those mutations that improve the aptitude of chromosomes will be selected. The defined mutation operators are: 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 and population size as a constant this involves the replacement of parents by the children chromosomes in generations with the exception to remain with elite chromosomes. The roulette selection is the criteria chosen for this.

The initiation of chromosomes is based on the received tasks; each chromosome is initiated with the tasks of the new case. For each task of the chromosome a sequential search is made in the case and assigns the task to an existing agent so that this association is maintained for the remaining tasks.

5 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 simulates 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), Mode 4 – The whole planning with CBR. In the first case (Test 1) a list of 500 records were entered within a period of 120 minutes and the second case (Test 2) had 1500 records into the system within 210 minutes. Previously it had a memory of 700 cases, based on cases where values were altered randomly following a normal distribution.

Table 1 Results obtained during the simulation

	Number of cases not processed in time		Benefits obtained	
Plan	Test 1	Test 2	Test 3	Test 4
Mode 1	5	25	1100	4100
Mode 2	4	20	1350	4500
Mode 3	2	9	1640	5060
Mode 4	1	5	1720	5290

Table 1 shows that the qualitative leap occurs in both cases by the introduction in the third mode of simulation, with 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.

The benefits obtained in different modes are even more significant. 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 without planning mode. 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, as the above chart shows, 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.

6 Conclusions and Future Work

Throughout this article a model of VO with a mechanism for planning and distributing tasks in a dynamic environment is presented. A genetic algorithm is introduced as a mechanism for the exploration of the 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.

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

References

1. Trippi, R.R., Ash, A.W., Ravenis, J.V.: A mathematical approach to large scale personnel assignment. Computers & Operations Research 1(1), 111–117 (1974)
2. Zülch, G., Rottinger, S., Vollstedt, T.: A simulation approach for planning and reassigning of personnel in manufacturing. International Journal of Production Economics 90, 265–277 (2004)
3. Durfee, E.H., Lesser, V.R., Corkill, D.D.: Trends in Cooperative Distributed Problem Solving. IEEE Transactions on Knowledge and Data Engineering 1(1), 63–83 (1989)
4. Shin, K., Corder, S.: Implementing the ford-fulkerson labeling algorithm with fixed-order scanning. Computers & Operations Research 19(8), 783–787 (1992)
5. Esteva, M., Rodríguez, J., Sierra, C., Garcia, P., Arcos, J.: On the formal specific ations of electronic institutions. In: Dignum, F.P.M., Cortés, U. (eds.) AMEC 2000. LNCS (LNAI), vol. 2003, pp. 126–147. Springer, Heidelberg (2001)
6. Kolodner, J.: Case-Based Reasoning. Morgan Kaufmann, San Francisco (1993)
7. Corchado, E., Assumpcio, M., MBorrajo, M.L.: A maximum likelihood Hebbian learning-based method to an agent-based architecture. Int. Journal of Computer Mathematics. 86(10-11), 1760–1768 (2009)
8. Martín, Q.: Investigación Operativa. Prentice-Hall, Englewood Cliffs (2003)
9. Menasce, D.A.: Trade-offs in designing Web clusters. IEEE Internet Computing 6(5), 76–80 (2002)

10. Yu, J., Buyya, R.: Scheduling scientific workflow applications with deadline and budget constraints using genetic algorithms. Scientific Programming 14(3), 217–230 (2006)
11. Rodríguez, P.: Discusión y Análisis de la metaheurística SN. Depto. Investigación Operativa, InCo, FI, UdelaR. Reporte Tecnico 03-02 (2003)
12. Seda, M.: Mathematical Models of Flow Shop and Job Scheduling Problems. Worl Academy of Science, Engineering and Technology (31), 122–127 (2007)
13. Wan, S., Yang, F., Izquierdo, E.: Lagrange multiplier selection in wavelet-based scalable video coding for quality scalability. Signal Processing: Image Communication 24(9), 730–739 (2009)
14. Pacheco, J.A., Casado, S.: Estudio Comparativo de Diferentes Metaheurísticas para la Resolución del Labor Scheduling Problem. Estudios de Economia Aplicada 21(3), 537–557 (2003)
15. Shi, C., Lu, J., Zhang, G.: An extended Kuhn-Tucker approach for linear bilevel programming. Applied Mathematics and Computation 162(1), 51–63 (2005)
16. Andres, C., Miralles, C., Pastor, R.: Balancing and scheduling tasks in assembly lines with sequence-dependent setup times. European Journal of Operational Research 187(3), 1212–1222 (2008)
17. Kim, K.H., Park, Y.: A crane scheduling method for port container terminals. European Journal of Operational Research 156(3), 752–768 (2004)
18. Porto, S., Kitajima, J.P., Ribeiro, C.: Performance evaluation of a parallel tabu search task scheduling algorithm. Parallel Computing 26(1), 73–90 (1996); Steinhofel, K., Albrecht, A., Wong, C.K.: Two simulated annealing-based heuristics for the job shop scheduling problem. European Journal of Operational Research 118(3), 524–548 (1999)
19. Taillard, E.: Parallel Taboo Search Technique for the Job shop Scheduling Problem. Journal on Computing Science 6, 108–117 (1994)
20. Kolonko, M.: Some new results on simulated annealing applied to the job shop scheduling problem. European Journal of Operational Research 113(1), 123–136 (2009)
21. Kabadi, S., Punnen, A.: A strongly polynomial simplex method for the linear fractional assignment problem. Operations Research Letters 36(4), 402–407 (2008)