Adaptive Approach Heuristics for The Generalized Assignment Problem

Helena Ramalhinho Lourenço[1,2]

Daniel Serra[1,3]

May, 1998

Abstract

The Generalized Assignment Problem consists in assigning a set of tasks to a set of agents with minimum cost. Each agent has a limited amount of a single resource and each task must be assigned to one and only one agent, requiring a certain amount of the resource of the agent. We present new metaheuristics for the generalized assignment problem based on hybrid approaches. One metaheuristic is a *MAX-MIN* Ant System (*MMAS*), an improved version of the Ant System, which was recently proposed by Stutzle and Hoos to combinatorial optimization problems, and it can be seen has an adaptive sampling algorithm that takes in consideration the experience gathered in earlier iterations of the algorithm. Moreover, the latter heuristic is combined with local search and tabu search heuristics to improve the search. A greedy randomized adaptive search heuristic (GRASP) is also proposed. Several neighborhoods are studied, including one based on ejection chains that produces good moves without increasing the computational effort. We present computational results of the comparative performance, followed by concluding remarks and ideas on future research in generalized assignment related problems.

Keywords: metaheuristics, generalized assignment, local search, GRASP, tabu search, ant systems.

[1] Department of Economics and Management, Universitat Pompeu Fabra, R. Trias Fargas 25-27, 08005 Barcelona, Spain

[2] e-mail: ramalhin@upf.es

[3] e-mail: serra@upf.es

1. Introduction

The **G**eneralized **A**ssignment **P**roblem (GAP) considers the minimum cost assignment of n jobs to m agents such that each job is assign to one and only one agent subject to capacity constraints on the agents.

GAP has several applications in areas like computer and communication networks, location problems, vehicle routing and machine scheduling. Our initial interest in this problem came from two applications, resource assignment problems and pure integer capacitated plant location problems.

The aim of this paper is to present several adaptive approach heuristics to solve the GAP, and the respective computational results. These heuristics can be embedded in a general framework with three steps; in the first step, a solution is constructed following a greedy randomized or an ant system approach; in the second step, a local search is applied to improve these initial solution, a descendent local search and tabu search are proposed; the last step consists in updating a set of parameters. The three steps are repeated until a stopping criterion is verified. The choices made in each step lead to different heuristic methods.

The paper is organized as follows: first, we present the GAP and a review of the methods proposed to solve it. In the next section, we describe the general framework of the adaptive approach heuristics. In section 4, we focus on local search methods, describing the descendent local search and the tabu search. In section 5, we describe the computational experiments to evaluate the proposed heuristics, present the computational results and perform a comparison with other methods. Section 6 concludes with general remarks on this work and directions of future research.

2. The Generalized Assignment Problem

The GAP consists in assigning a set of tasks to a set of agents with minimum cost, such that each agent has a limited amount of a single resource and each task must be assigned to one and only one agent requiring a certain amount of the resource of the agent. This problem is well-know, for an extended review see, for example: Martello and Toth[1], Cattrysse and Van Wassenhove[2] and P. Chu[3]. Fisher, Jaikumar and Van Wassenhove[4] proved that the problem is NP-hard. Moreover, the problem of finding if there exits a feasible solution is NP-Complete. Osman[5] presented a survey in many real-life applications.

The Generalized Assignment Problem can be formulated as an integer program, as presented next. Consider the following notation:
I : set of tasks (i=1, ..., n) .
J : set of agents (j=1, ..., m) .
a_j = resource capacity of agent j .
b_{ij} = resource needed if task i is assigned to agent j .
c_{ij} = cost of task i if assigned to agent j .

The variables are : x_{ij} = 1, if task i is assigned to agent j; 0, otherwise.

$$(1)\quad \min f(x) = \sum_{j=1}^{m}\sum_{i=1}^{n} c_{ij}x_{ij}$$

$$s.a$$

$$(2)\quad \sum_{i=1}^{n} b_{ij}x_{ij} \le a_j \quad j = 1,\ldots,m$$

$$(3)\quad \sum_{j=1}^{m} x_{ij} = 1 \quad i = 1,\ldots,n$$

$$(4)\quad x_{ij} \in \{0,1\} \quad i = 1,\ldots,n;\ j = 1,\ldots,m$$

Constraints (2) are related to the resource capacity of the agents, constraints (3) guarantee that each task is assign to one agent, and since the variables are binary then each task is assigned to one and only one agent.

Several exact algorithm for GAP have been proposed by Ross and Soland[6], Martello and Toth[1], Fisher, Jaikumar and Van Wassenhove[4], Guinard and Rosein[7], and Karabakal et al.[8]. Recently, Salvelsberh[9] presented a branch-and-cut algorithm that employs both column generation and branch-and-bound to obtain the optimal solution to a set partitioning formulation of the GAP. The author mentioned that problems with 20 agents and 50 jobs can be solved from 210 to 1160 seconds.

Also, several heuristics have been proposed to solve the GAP. Amini and Racer[10] presented a variable-depth-search heuristic motivated by the work of Lin and Kernighan to the Traveling Salesman Problem and they offered rigorous statistical experiment and analysis of the solution methods; Trick[11] proposed an approximation algorithm; Cattrysse, Salomon and Van Wassenhove[12] formulated the problem as a Set Partitioning problem and proposed an heuristic based on column generation techniques; Osman[5] presented a comparative performance of algorithms based on tabu search and simulated annealing techniques; Wilson[13] presented a genetic algorithm to restore feasibility to a set of near-optimal solutions, and then improve the best solution found by local search; Chu and Beasley[3] also presented a genetic algorithm for the GAP that tries to improve feasibility and optimality simultaneously. Laguna et al.[14] proposed a tabu search algorithm based on ejection chains to an extension of GAP, the multilevel generalized assignment problem.

3. Adaptive Approach Heuristic

In this section with present the general framework and the principal aspects of the adaptive heuristics proposed to solve the GAP. These adaptive heuristics are based on two metaheuristics approaches to solve combinatorial optimization problems: the Greedy Randomized Adaptive Search Heuristic, GRASP, Feo and Resende[15] and the *Max-Min* Ant System (MMAS), Stutzle and Hoos[16,17]. Both techniques include a step where a local search method is applied. The local search methods are presented in the next section, since they present some innovation aspects.

3.1 General Framework

The adaptive approach heuristics proposed to solve the generalized assignment problem can be described in a general framework composed by three steps, that are repeated iteratively until some stopping criteria is verified:

Step 1: Construct a solution in a greedy fashion.
Step 2: Apply a local search method.
Step 3: Update the parameters (if any).

The approach adopted for the first step is based on greedy heuristics. A greedy heuristic constructs a solution as follows: at each step, a next task to be assigned is chosen; then, the next choice is the agent to which the chosen task is assigned. This procedure is repeated until all tasks have been assigned to an agent.

We propose two heuristics for this first step: A Greedy Randomized Adaptive Heuristic (GRAH) and a Ant System Heuristic (ASH). The basic difference between the basic greedy heuristic and the GRAH and ASH is the choice of the agent to assign the chosen task. For the basic greedy heuristic, the choice is deterministic and based on a greedy function, for example the cost function. For the GRAH, the choice is a probabilistic bias to a probability function, which do not depend on the iteration of the general framework. For the ASH, the choice is also a probabilistic bias on a desirability function. This function is updated at each iteration in a reinforcement learning way, keeping track of the features of the good solutions found in the search.

In the second step, two local search are proposed, a descendent local search and a tabu search approach. The step is only applied if there are parameters that have to be actualized at the end of each iteration, which is the case for the ant system heuristic. Each method differs in the way choices are made at each step of the above framework.

We will admit infeasible solutions with respect to capacity constraints, i.e. the total resource required by the tasks assigned to some agents may exceed the capacity of these ones. Infeasible solutions will penalize the objective function.

The principal reason for admitting infeasible solutions is that, for some solutions close to the optimal one, the capacity of the agents will be almost full, therefore any neighbor obtained by interchange or reassign tasks will be or infeasible or a “bad” neighbor. Allowing these extra solutions usually provides escape routes out of local optima. This approach is quite common in implementations of metaheuristics, and is often very effective. (see Johnson et al.[18])

The penalty function is as follows:

$f'(x) = \sum_{j=1}^{m}\sum_{i=1}^{n} c_{ij}x_{ij} + \alpha \sum_{j=1}^{m} \max\left\{0, \sum_{i=1}^{n} b_{ij}x_{ij} - a_j\right\}$ where $\alpha > 0$ is a parameter, representing the cost of using one unit of overloaded capacity.

If a solution is not feasible the second term will be positive and therefore the search will look to feasible solution. The parameter α can be increased during the run to penalize infeasible solutions, and drive the search to feasible ones.

3.2 Greedy Randomized Adaptive Procedure

The first method proposed to solve the GAP is based on the greedy randomized adaptive search heuristic (GRASP)[15]. GRASP is an iterative randomized sampling technique, with two phases. The first phase consists in a greedy randomized adaptive heuristic that constructs an initial solution. It is called adaptive because the greedy function takes in account previous decisions in the construction when considering the next choice. The second phase consists in an improvement phase which usually corresponds to a local search method. GRASP has been successfully applied to many combinatorial optimization problems. As it can be seen, the GRASP fits in the general framework proposed in previous section. For a list of several applications of GRASP see, Resende, Pitsoulis and Pardalos[19] and other papers in site: http://www.research.att.com/~mgcr/.

In this section we will only describe the greedy randomized adaptive heuristic, i.e. the first phase. The local search will be explained in the next section.

At each iteration of the Greedy Randomized Adaptive Heuristic (**GRAH**) one task is assigned to an agent. The heuristic finishes when all tasks have been assigned. The GRAH can be described as follows:

1. Let $S_j = \varnothing \quad \forall j = 1,\ldots,m$. ($S_j$ is the set of task assigned to agent *j*)
2. Construct the RCL (restricted candidate list) of agents for each task, L_i, such that $L_i = \left\{ j : c_{ij} \le c_{\max} \right\}$, $c_{\max}$ is a parameter that limits the dimension of the restricted list (if $c_{\max} = \max_{\forall i,j} c_{ij}$ all agents will be included).
3. Consider any order of the tasks, *i=1*.
4. While (not all tasks have been assigned) repeat
   4.1. Choose randomly an agent *j** from L_i following the probability function that depends on the resource of agent *j* and the resource need by task *i*: $p_{ij} = \dfrac{a_j / b_{ij}}{\sum_{l \in L_i} a_l / b_{il}}, \quad j \in L_i$. The agent with minimal cost has greater probability to be chosen.
   4.2. Assign task *i* to agent $j^*: S_{j^*} = S_{j^*} \cup \{i\}$. Let *i=i+1* and if $\sum_{i \in S_{j^*}} b_{ij^*} > a_{j^*}$ remove *j** from any list. Repeat step 4. (Note that the capacity constraint can be violated).
5. Let $x_{ij} = 1$ if $i \in S_j$; $x_{ij} = 0$, otherwise. Calculate the value of the penalty function for the solution, *f'(x)*.

In reality, we are looking initially for a feasible solution. Therefore, we are interested in assigning a task to agent if this task uses a small amount of the resource of the agent, i.e. small b_{ij} / a_j. For each task, we order the agents in function of p_{ij}, in decreasing order. The task can be assigned to any agent following the probability function p_{ij}, if there is available capacity. If not, the task is assigned to the first agent in the above order with left capacity. If all agents are full, the assignment is random. Note that the solution obtained can be infeasible with respect to the capacity constraints.

3.3 The Ant System

The GRASP procedure in the previous chapter can be seen has an multi-start local search, but, instead of considering random initial solution, a greedy randomized heuristic is used to try to find better initial solutions than the random ones.

Recently, Stutzle and Hoos[16,17] have proposed a improved version of the Ant System, that can also be seen has a adaptive sampling algorithm, but takes in consideration the experience gathered in earlier iterations of the algorithm, designated by *MAX-MIN* Ant system (*MMAS*). Moreover, combining MMAS with local search, Stutzle and Hoos[16,17,20] were able to find very good solutions to the Traveling Salesman Problem, Quadratic Assignment Problem and Flow-Shop Scheduling Problem.

Next, we will propose a MMAS with Local Search approach for the GAP, which can be seen as an extension of the GRASP heuristic of the previous section.

The Ant System, introduce by Colorni, Dorigo and Maniezzo[21,22], is a cooperative search algorithm inspired in behavior of real ants. Ants lay down in some quantity an aromatic substance, know as pheromone, in their way to food. An ant chooses a specific path in correlation with the intensity of the pheromone. The pheromone trail evaporates over time if no more pheromone in laid down by other ants, therefore the best paths have more intensive pheromone and higher probability to be chosen.

The Ant System approach associates pheromone trails to features of the solutions of a combinatorial problem, which can be seen as a kind of adaptive memory of the previous solutions. Solutions are iteratively constructed in a randomized heuristic fashion biased by the pheromone trails, left by the previous ants. The pheromone trails, τ_{ij}, are updated after the construction of a solution, enforcing that the best features will have a more intensive pheromone.

The *MAX-MIN* ant system differs from the Ant System in the following way: only the best ant updates the trails in every cycle. To avoid stagnation of the search, i.e. the ants always choose the same path, Stutzle[20] proposed a lower and upper limit to the pheromone trail, $\tau_{\min}$ and $\tau_{\max}$, respectively.

Next, we propose a *MAX-MIN* Ant System with Local Search for the GAP, based on the work of Stutzle[20] for the Flow-Shop Scheduling Problem.

Let τ_{ij} be the desirability of assigning a task *i* to an agent *j*. Initially let $\tau_{ij} = \frac{1}{c_{ij}}$ (for generalized assignment problems with maximization, we have use $\tau_{ij} = c_{ij}$). The cheaper the assignment of the task *i* to agent *j* is, the more desired is the assignment.

Firstofall, let us explain how to construct a solution using only one ant, as suggested by Stutzle[20]. This heuristic will be designated by Ant System Heuristic. The tasks are assigned to

the agents in a greedy way like in the Greedy Randomized Adaptive Heuristic (except for step 4.1), where the assignment is done biased by the τ_{ij}. A task *i* is assigned to a particular agent *j* in the following way:

1. With probability p_0, choose the agent *j** with maximal value of τ_{ij}.
2. With probability $1 - p_0$, choose the agent *j** according to the following probability distribution:

$$p_{ij} = \begin{cases} \dfrac{\tau_{ij}}{\sum_{l \in L_i} \tau_{il}} & \text{if } j \in L_i \\ 0 & \text{otherwise} \end{cases}$$

This assignment constitutes the first step in the general framework, followed by a local search that tries to improve this initial solution. Afterwards, in the third step of the general framework, the pheromone trails are updated in the following way:

$\tau_{ij}^{new} = \rho \tau_{ij}^{old} + \Delta_{ij}$, where ρ, $0 < \rho < 1$, is the persistence of the trail, i.e. $1 - \rho$, represents the evaporation.

Also, the updated amount is $\Delta_{ij} = \begin{cases} \tau_{max} \times Q & \text{if task } i \text{ is assign to agent } j \text{ in the solution} \\ 0, & \text{otherwise} \end{cases}$

where $Q = \begin{cases} 0.01, & \text{if the solution is infeasible;} \\ 0.05, & \text{if the solution is feasible.} \end{cases}$

Note that, if the solution is feasible, the pheromone trail has a bigger increment, trying to give greater probability to feasible assignments.

Moreover, $\tau_{min} \le \tau_{ij} \le \tau_{max}, \forall i, j$, so these limits must be imposed if the updated pheromone falls outside the above interval.

Next, we present the local search methods for the second step of the adaptive approach heuristics.

4. Local Search Methods

Local search methods are improvement search techniques, extensively used to obtain good solution for combinatorial hard problems. In order to derive a local search method, it is necessary to define a neighborhood structure, that is a function that associates a set of solutions *N(x)* with each solution *x*. The neighborhood is usually obtained by specific modifications on *x*, called a moves.

The local search starts with a initial solution and searches the neighborhood defined before for one solution with some characteristics, as for example the one with lower cost. Then this neighbor solution replaces the current solution if it verifies some properties that depend on the acceptance strategy as, for example, if it has a lower cost than the current solution. The search

continues until a stopping criterion is verified. The algorithm returns the best solution found with respect to the cost function.

4.1 Neighborhoods

We present two neighborhoods for the GAP, a simple shift neighborhood where a task is reassigned to a new agent and an ejection chain neighborhood where more than one agent is reassigned to new agents. The shift neighborhood is special case of the λ-generation mechanism proposed to Osman[5], where $\lambda = 1$, with subset size (0,1) or (1,0). The ejection chain neighborhood is based on the work of Laguna et al.[14].

The move in the shift neighborhood consists in removing a task from one agent and assign it to another agent. The size of the neighborhood is $n(m-1)$. Note that we should start by considering for removing tasks in overloaded agents. The task can be reassigned to a overloaded agent.

The shift neighborhood can be obtained by the following procedure, Neighborhood (x, *flag*):

1. Order the agents by the amount of capacity over the maximum in decreasing order. Let $j=1$.
2. Consider any order of the tasks assign to agent j. Let $i_j = 1$.
3. Remove i_j from j, $S_j = S_j - \{i_j\}$.
4. Assign i_j to the another agent, not yet considered, starting in the last agent in list, (neighbor x' of x).
5. Calculate the value of the penalty function for x', $f'(x')$. If $f'(x') <= f'(x)$, go to step 8.
6. Let $i_j = i_j + 1$ and repeat steps 3 and 4, until all tasks of j have been considered.
7. Let $j=j+1$, and repeat from step 2, until all agents have been considered.
8. Let $x=x'$, *flag=true* and stop.

In preliminary tests, these neighborhoods were able to obtain feasible solutions when starting from an infeasible one. However, the local optimal solutions were not very good. This lead us to define a more complex neighborhood, that we present next.

The ejection chain neighborhood is a variable depth procedure which consists in move more than one task from the current agent to a new agent. The neighborhood structure based on ejection chains was introduced by Glover[23], and have been applied to several problems, including an extension of the GAP, Laguna et al.[14].

This second neighborhood structure is more complex than the shift neighborhood, but leads to a more powerful and efficient search without increasing significantly the computational running time.

The ejection chain neighborhood can be obtained by the application of the following two types of moves:

Move A: Remove a task i from an agent (agent j), then insert this task i in a different agent (agent w).

Move B: Remove a task *i* from an agent (agent *j*), then insert this task *i* in a different agent (agent *w*). After, remove a task *k* from agent *w* (*w* different from *j*) and insert task *k* in another agent (different from *w*, but it can equal of *j*).

The ejection chain neighborhood of a solution can be obtained in a similar way as it was done for the shift neighborhood, but move of type B is only applied if the move of type A was not successful. Note also that the number of neighbors is of order $O(n^2m^2)$, a larger number than the shift neighborhood.

4.2 Descendent Local Search

We designated by descendent local search the local search method with steepest descendent and first improvement strategy. This means that the first cost improvement neighbor solution found, becomes the new current solution. This method stops at a local optimal solution with respect to the neighborhood structure chosen. This can be the main drawback of this method, since it is unable of overcome a local optimal solution. The tabu search method presented in the next section has a different acceptance strategy and other features designed for avoiding being trapped at a bad local optimum.

The main steps of the descendent local search are:
1. Obtain an initial solution x (for example, using the GRAH).
2. Let *flag=false*;
3. Neighborhood(*x*);
4. If *flag=false*, stop (a local optimum was found), otherwise repeat step 3.

If the heuristic finishes with a infeasible solution, apply a 2-opt local search with the neighborhood: interchange tasks between agents considering only moves that reduce the overloaded capacity. After a feasible solution is obtained, we can apply the same 2-opt local search considering only feasible solutions, i.e. verifying the following conditions:
Let (i,k) and (j,l) be pairs of tasks and agents respectively, such that $x_{ij} = x_{kl} = 1$. If $c_{il} + c_{kj} < c_{ij} + c_{kl}$, $\sum_{s=1}^{n} b_{sj}x_{sj} - b_{ij} + b_{kj} \le b_j$ and $\sum_{s=1}^{n} b_{sl}x_{sl} - b_{kl} + b_{il} \le b_l$, then let $x_{il} = x_{kj} = 1$ and $x_{ij} = x_{kl} = 0$.

We are now on conditions to present the GRASP method which consist on one of the approaches proposed to solve the GAP:
1. While a stopping criterion not satisfied:
   1.1. Construct a solution (*x*) using the Greedy Randomized Adaptive Heuristic. In the first iteration initialize x_b, the best solution.
   1.2. Apply local search(*x*)
   1.3. If *x* is feasible, and $f(x)<f(x_b)$ let $x_b=x$.
2. Return the best solution found, x_b.

The next method for the GAP, also based on the general framework presented before is the procedure *MAX-MIN* Ant System with Local Search (MMAS), and can be described as follows:

1. Initialize the pheromone trails and parameters
2. While (termination condition not met)
   2.1. Construct a solution using the Ant System Heuristic. In the first iteration initialize x_b, the best solution.
   2.2. Apply local search(x)
   2.3. If x is feasible, and $f(x)<f(x_b)$ let $x_b=x$ and update the pheromone trails.
3. Return the best solution found, x_b.

In both methods, the stopping criterion applied consists in a maximum number of iterations.

Note that the main difference in the above methods is the way the initial solutions are constructed, i.e. the first step in the general framework of the adaptive approach heuristics. The GRASP follows a random approach by means of a random sampling over solutions constructed in a greedy fashion, which can be seen as a diversification strategy. Meanwhile, the MMAS constructs the initial solutions using adaptive and cooperative memory, that can be seen as an intensification strategy.

4.3 Tabu Search

Tabu search was originally proposed by Glover[24], and since then these metaheuristics have been subject to extensive studies and applied to several optimization problems with great success. Tabu search can be described as a intelligent search that uses memory to drive the search out of local optimal solutions and find good results. For a survey in tabu search see Glover and Laguna[25].

Our motivation to apply tabu search to GAP was the excellent results obtained by Laguna et al.[14] for a multilevel generalized assignment problem. They also presented computational results to the GAP and were able to obtain always the optimal solution for test problems with 5 agents and 25 task in less than 1.30 seconds.

The tabu search can be briefly described as follows:

1. Suppose we have a initial solution x.
2. While the stopping criteria is not met do:
   2.1. Generate the candidate list of moves/neighbors;
   2.2. Choose the best neighbor not tabu or verifying the aspiration criteria, x';
   2.3. Update the current solution, $x=x'$.
3. Output the best solution found.

The basic ingredients of the tabu search are: the neighborhood, the tabu list and it size, the aspiration criteria and the stopping criteria. Next we will describe these aspects in detail for the GAP.

It can be easily observed that the ejection chain neighborhood has a large number of neighbors. Some of them lead to very bad solutions. Therefore, to avoid spend a large amount of time evaluating the neighborhood, a candidate list strategy is used to restrict the number of

solutions examined at each iteration of the tabu search. The candidate list strategy implemented uses context information to limit the search to those moves that are more likely to improve the current solution.

The problem-specific candidate list strategy can be described as follows: a task or a pair of tasks are considered for moving if one of the following situations occur. Let *x* be the current solution and *x'* a neighbor solution:

- A task *u* is considered for moving from a agent *p* to an agent *k* if $c_{up} > c_{uk}$;
- A pair of tasks *u, l* are considered for moving if $c_{up} + c_{lk} > c_{uk} + c_{lq}$;
- A task *u* is considered for moving from a agent *j* to an agent *k* if $\sum_{i=1}^{n} b_{ip} x_{ip} > a_p$ and $\sum_{i=1}^{n} b_{ik} x'_{ik} \le a_k$, where $x'_{ij} = x_{ij} \forall i \ne u, j \ne k, p; x'_{up} = 0, x'_{uk} = 1$.
- A pair of task *u, l* are considered for moving if $\sum_{i=1}^{n} b_{ip} x_{ip} > a_p$, $\sum_{i=1}^{n} b_{ik} x'_{ik} \le a_k$ and $\sum_{i=1}^{n} b_{iq} x'_{iq} \le a_q$, where $x'_{ij} = x_{ij} \forall i \ne u, l; j \ne k, p, q; x'_{up} = 0, x'_{uk} = 1, x'_{lk} = 0, x'_{lq} = 1$.

A tabu attribution is related to the move of a task from one agent to another agent, i.e. suppose a task *i* is assigned to an agent *j* in the current solution, and for the accepted neighbor, this task is reassigned to agent *k*, then for a certain number of iterations, the tabu tenure or the size of the tabu list, is forbidden to assign again task *i* to agent *j*. The tabu list was implemented as a matrix *n*m*, such that the entry *(i,j)* contains the iteration number where the task *i* was removed from agent *j*, therefore in the next "tabu tenure" iterations the move "assign task *i* to agent *j*" is tabu.

An aspiration criteria is considered, that overrules the tabu attribution if the move leads to the best solution found so far. And, the tabu search stops after a maximum number of iteration.

As we have done for the descendent local search, now we have two more methods for the GAP, GRASP/TABU and MMAS/TABU that can be described as before, but instead of using the previous local search method, the tabu search is applied.

5. Computational Experiment

In the section, we will present the computational experiment and the results obtained. We have follow the guidelines proposed by Barr et al.[26]. The computational experiment was designed with three main objectives:

- Gain understanding of the behavior of the different proposed methods, based on the general framework of the adaptive approach heuristics;
- Compare the two methods proposed for the first step of the general framework: greedy randomized adaptive heuristic versus the ant system heuristic.
- Compare the best methods described in this work with other techniques and methodology proposed to solve the generalized assignment problem.

All methods described were coded in Fortran, and were tested in a set of problems ranging from 5 agents/15 jobs to 10 agents/60 jobs. These test problems are publicly available from the OR library (http://www.ms.ic.ac.uk/info.html) and have been used also by other authors in their computational experiments, Osman[5], Cattrysse, Salomon and Van Wassenhove[12], Chu and Beasley[3]. The set of test problems can be divided in two groups: easy (gap 1 to gap 6) and difficult (gap7 to gap12). These set problems are of maximization form of GAP, so we have converted them in minimization form. All numerical tests were carried on a PC-Pentium with 16.0 MB RAM.

The performance measures considered are:

- Quality solution measured as the percentage deviation to optimal.
- Computational time: total running time and the time to best found solution (measured at the end of one iteration of the general framework).

The factors that can influence the behavior of a method and their results are:

- Problem specific: number of agents (n); number of task (m); resource capacity of the agents (a_j), cost of the overloaded capacity (α).
- First Step: greedy randomized adaptive heuristic; ant system heuristic.
- Second Step: neighborhood, search strategy: escendent local search and tabu search.
- Stopping criteria: number of total iterations (NTI) and iterations of the tabu search (NITB).
- Other parameters: size of the tabu list (STL), ant system parameters (τ_{min}, τ_{max}, ρ and p_0).

If we want to consider all the above factors, the experimentation would be quite extensive. Thus to minimize the computational effort some of the above factors are chosen a priori based on previous experiments for the GAP or on preliminary computational results.

Consider the following values for the parameters for the ant system heuristic, that were fixed in preliminary runs: $\tau_{min} = 0.1 \times \min_{\forall i,j} \tau_{ij}$ and $\tau_{max} = n \times \max_{\forall i,j} \tau_{ij}$, $\rho = 0.75$ and $p_0 = \frac{n-m}{n} \times 0.8$.

5.1 Comparison between different approaches

The main issue for these initial tests is to understand the behavior of the different methods based in the same general framework and on different approaches. The adaptive search heuristics considered are the following ones:

MMAS : ant system heuristic and descendent local search with ejection chains neighborhood.

GRASP : greedy randomized adaptive heuristic and descendent local search with ejection chains neighborhood. This version is a GRASP method.

ASH+TS : ant system heuristic and tabu search with restricted ejection chains neighborhood.

GRAH+TS : greedy randomized adaptive heuristic and tabu search with restricted neighborhood ejection chains.

ASH+LS+TS : ant system heuristic, descendent local search with shift neighborhood, and tabu search with restricted neighborhood ejection chains.

GRAH+LS+TS : greedy randomized adaptive heuristic, descendent local search with shift neighborhood, and tabu search with restricted ejection chains neighborhood.

ASH+LS+CTS : ant system heuristic, descendent local search with shift neighborhood, and tabu search with ejection chains neighborhood (search in the complete neighborhood, best-improvement).

In the three last methods, before applying the tabu search method, we apply a simple descendent local search method with shift neighborhood, since most of the solutions obtained in first step are infeasible and the descendent local search with shift neighborhood usually finds a feasible one in a short time, which allows the tabu search to start from a better solution. With the last method, we will try to analyze the effect of using or not a restricted candidate list.

Table 1: Average percentage deviation to optimal for 5 runs of the adaptive search heuristics

prob.	na*nt	optimal	MMAS	GRASP	ASH+TS	GRAH+TS	ASH+ LS+TS	GRAH+ LS+TS	ASH+ LS+CTS
gap7-1	8*40	942	0.08%	0.04%	0.00%	0.00%	0.00%	0.00%	0.00%
gap7-2	8*40	949	0.00%	0.02%	0.00%	0.00%	0.00%	0.00%	0.00%
gap7-3	8*40	968	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
gap7-4	8*40	945	0.00%	0.17%	0.00%	0.00%	0.00%	0.00%	0.00%
gap7-5	8*40	951	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
gap8-1	8*48	1133	0.32%	0.34%	0.14%	0.09%	0.05%	0.25%	0.28%
gap8-2	8*48	1134	0.07%	0.11%	0.00%	0.02%	0.00%	0.02%	0.00%
gap8-3	8*48	1141	0.210%	0.280%	0.105%	0.140%	0.070%	0.123%	0.175%
gap8-4	8*48	1117	0.143%	0.269%	0.054%	0.125%	0.000%	0.036%	0.018%
gap8-5	8*48	1127	0.248%	0.266%	0.106%	0.089%	0.089%	0.106%	0.142%
gap9-1	10*30	709	0.000%	0.028%	0.000%	0.000%	0.000%	0.000%	0.000%
gap9-2	10*30	717	0.223%	0.223%	0.000%	0.056%	0.000%	0.056%	0.056%
gap9-3	10*30	712	0.000%	0.000%	0.000%	0.000%	0.000%	0.000%	0.000%
gap9-4	10*30	723	0.111%	0.028%	0.000%	0.000%	0.000%	0.000%	0.000%
gap9-5	10*30	706	0.057%	0.000%	0.000%	0.000%	0.000%	0.000%	0.000%
gap10-1	10*40	958	0.000%	0.000%	0.000%	0.000%	0.000%	0.000%	0.000%
gap10-2	10*40	963	0.042%	0.104%	0.000%	0.000%	0.000%	0.021%	0.000%
gap10-3	10*40	960	0.229%	0.250%	0.125%	0.125%	0.063%	0.104%	0.104%
gap10-4	10*40	947	0.084%	0.169%	0.000%	0.000%	0.000%	0.000%	0.021%
gap10-5	10*40	947	0.211%	0.190%	0.063%	0.042%	0.000%	0.063%	0.042%
gap11-1	10*50	1139	0.018%	0.070%	0.000%	0.000%	0.000%	0.000%	0.000%
gap11-2	10*50	1178	0.000%	0.034%	0.000%	0.000%	0.000%	0.000%	0.017%
gap11-3	10*50	1195	0.000%	0.000%	0.000%	0.000%	0.000%	0.000%	0.000%
gap11-4	10*50	1171	0.051%	0.017%	0.000%	0.000%	0.000%	0.000%	0.000%
gap11-5	10*50	1171	0.034%	0.102%	0.000%	0.000%	0.000%	0.000%	0.000%
gap12-1	10*60	1451	0.055%	0.096%	0.000%	0.014%	0.000%	0.014%	0.014%
gap12-2	10*60	1449	0.055%	0.055%	0.000%	0.000%	0.000%	0.000%	0.000%
gap12-3	10*60	1433	0.000%	0.028%	0.000%	0.000%	0.000%	0.000%	0.000%
gap12-4	10*60	1447	0.041%	0.069%	0.000%	0.000%	0.000%	0.000%	0.000%
gap12-5	10*60	1446	0.028%	0.055%	0.000%	0.000%	0.000%	0.000%	0.028%
Average			**0.077%**	**0.100%**	**0.020%**	**0.023%**	**0.009%**	**0.026%**	**0.030%**

In this experiment, the following factors are prefixed: NTI =30, NITB=200, α=50, STL=10. Also, since all the above method performed very well in the easy test problems, we will present the results for the subset of large size problems, gap7 (8 agents, 40 jobs) to gap12 (10 agents, 60 jobs). For each test problem, we have performed 5 runs of each of the methods.

In Table 1 we present the average percentage deviation to optimal of 5 runs for each of the heuristics proposed. Firstofall, we observe that the best results were obtained by the AS+TS, GRAH+TS, AS+LS+TS and GRAH+LS+TS, i.e. when the tabu search was used in the second step of the general framework, and if combined with a local search method the results improve. MMAS and GRASP obtained the worst results, since there are many bad local optimal solutions. Therefore, using a tabu search approach made able to keep searching for good solutions. When a tabu search considering the complete neighborhood was used, ASH+LS+CTS, the quality of the solution did not improve. So, the use of restricted candidate lists play an important role in the search, helping finding good solution in significantly less time, see Tables 3 and 4. It can also be seen that the proposed heuristic perform very well, finding the optimal solution for many instances. For those in which the heuristics failed to reach the optimal, the solutions obtained are very close to optimality. In Table 2 we present the average quality solution for each set of the test problems. We can observe that the ASH+LS+TS performs better than the remaining heuristics, obtaining the optimal in all runs for all test problems in 4 of the 6 groups.

Table 2: Average percentage deviation to optimal of the adaptive search heuristics for the 6 group of problems.

prob.	na*nt	MMAS	GRASP	ASH+TS	GRAH+TS	ASH+LS+TS	GRAH+LS+TS	ASH+LS+CTS
gap7	8*40	0.017%	0.047%	0.000%	0.000%	0.000%	0.000%	0.000%
gap8	8*48	0.198%	0.251%	0.081%	0.092%	0.042%	0.106%	0.124%
gap9	10*30	0.078%	0.056%	0.000%	0.011%	0.000%	0.011%	0.011%
gap10	10*40	0.113%	0.143%	0.038%	0.033%	0.013%	0.038%	0.034%
gap11	10*50	0.021%	0.045%	0.000%	0.000%	0.000%	0.000%	0.003%
gap12	10*60	0.036%	0.061%	0.000%	0.003%	0.000%	0.003%	0.008%
Average		**0.077%**	**0.100%**	**0.020%**	**0.023%**	**0.009%**	**0.026%**	**0.030%**

Table 3: Average total running time in seconds of the adaptive search heuristics

prob.	na*nt	MMAS	GRASP	ASH+TS	GRAH+TS	ASH+LS+TS	GRAH+LS+TS	ASH+LS+CTS
gap7	8*40	94.0	120.7	111.1	139.5	113.7	125.3	342.0
gap8	8*48	172.0	232.3	172.8	214.5	178.9	205.9	576.1
gap9	10*30	54.6	74.9	72.8	81.3	74.0	79.2	209.6
gap10	10*40	141.4	172.2	146.8	160.2	137.9	150.3	468.3
gap11	10*50	256.6	334.6	140.8	192.9	144.1	163.5	878.9
gap12	10*60	427.0	530.4	238.5	349.9	242.9	320.4	1485.3
Average		**190.9**	**244.2**	**147.1**	**189.7**	**148.6**	**174.1**	**660.0**

In Table 3 and 4 we present the average total running CPU time and the average time to find the best solution for the 6 test problems. For all heuristics, the running increases with the ratio *m/n*, and also with the number of task. For the same number of global iterations of the general framework, the ant system heuristic (MMAS, ASH+TS, ASH+LS+TS) always takes less time than the greedy randomized adaptive heuristic (GRASP, GRAH+LS, GRAH+LS+TS). The computational time to find the best solution falls below the total running time, and again the ant system heuristic finds the best solution faster. However the difference between the ASH+TS, ASH+LS+TS and GRAH+LS+TS is not significant. The explanation for this

behavior is that the tabu search with the ejection chain neighborhood proceeds in a efficient way to find good solution.

Table 4: Average running time to best found solution, in seconds, of the adaptive search heuristics

prob.	na*nt	MMAS	GRASP	ASH+TS	GRAH+TS	ASH+LS+TS	GRAH+LS+TS	ASH+LS+CTS
gap7	8*40	35.8	42.8	19.7	56.6	22.5	21.4	92.4
gap8	8*48	80.3	88.8	68.1	96.5	85.3	78.3	225.9
gap9	10*30	18.8	22.2	19.0	10.7	15.3	13.4	55.8
gap10	10*40	45.5	85.4	35.7	51.2	37.5	34.5	159.7
gap11	10*50	84.1	98.3	33.8	64.3	28.4	33.9	278.2
gap12	10*60	137.5	211.9	59.3	91.7	59.7	95.0	379.7
Average		**67.0**	**91.5**	**39.3**	**61.8**	**41.5**	**46.1**	**198.6**

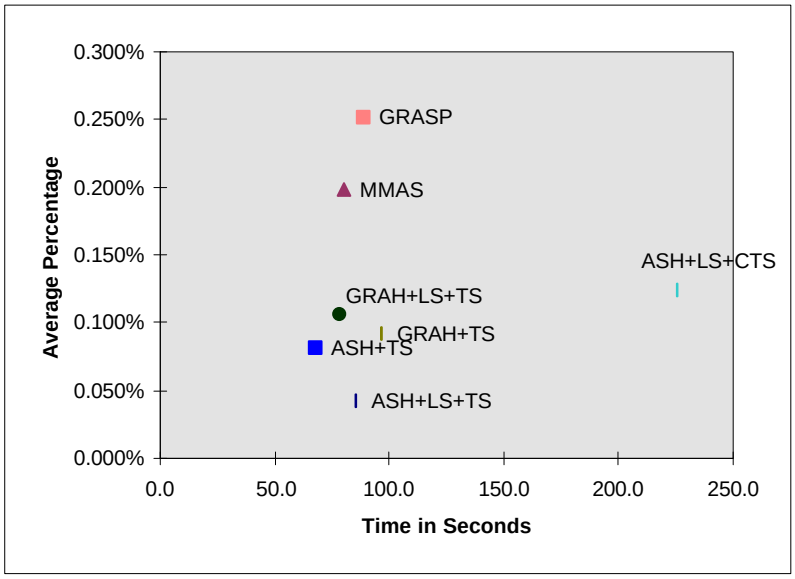

Figure 1: Average percentage deviation from optimal versus time for gap8 instances

To better understand the behavior of the several heuristics proposed we present two figures where we compare the tradeoff between solution quality and computational effort. For simplicity we present the average results for the test problem gap8, Figure 1, and gap10, Figure 2. It can be easily observed that the heuristics that obtain better results in terms of solution quality and computational time are the ASH+LS+TS, ASH+TS and GRAH+LS+TS in approximately this order, since these dominate the remaining ones. And, if we have to choose only one, our choice will be the ASH+LS+TS because obtains the best solution within a reasonable computational time.

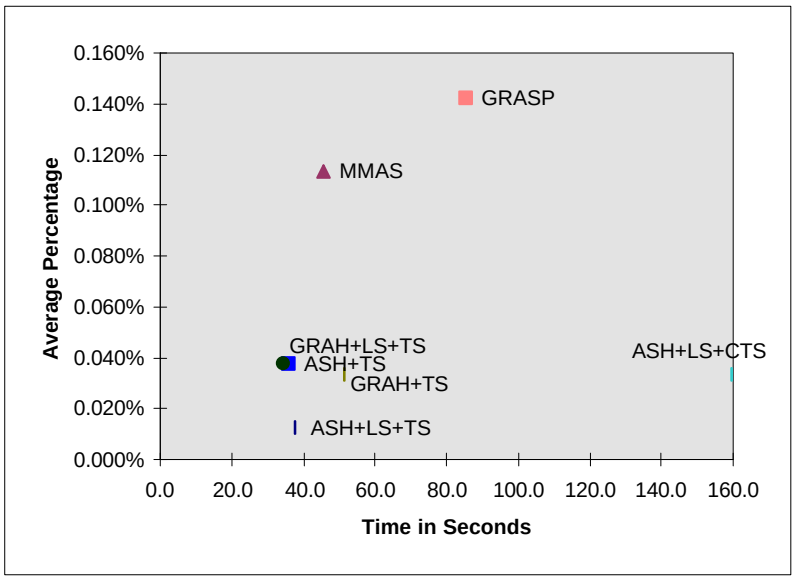

Figure 2: Average percentage deviation from optimal versus time for gap10 instances

5.2 Comparison between the adaptive approaches

A second issue that we would like to answer is related with the different approaches proposed for the first step. The greedy randomized adaptive heuristic is based on the use of randomization to obtain initial solutions in a greedy fashion and on this way diversify the search for a good solution. The other approach, based on the ant system, uses information of the good solution visited in previous iterations to construct a solution following also a greedy approach. We would like to analyze if there exists any difference on these two approach for the GAP, and specially, for some particular instances. Therefore, all the factors are kept constant, except for the two different heuristics proposed for the first step. We present the average results for the 6 test problems when the GRAH and the ASH were used in the first step of the general framework, and combined with the local search with ejection chain neighborhood, Figure 3, or with local search and tabu search, Figure 4. The results are presented by showing the tradeoff between solution quality and computational time. We can observe that when the ant system heuristic is used in the first step the method obtains better solutions in less time for most of the test problems.

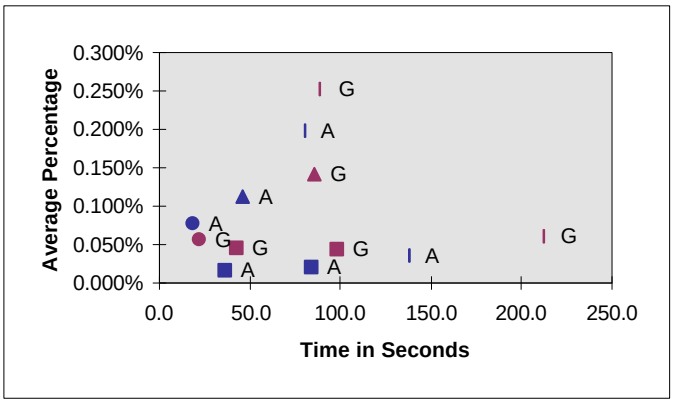

Figure 3: Average percentage deviation from optimal versus time of the MMAS (A) and GRASP (G) heuristics for gap7 to gap12 instances

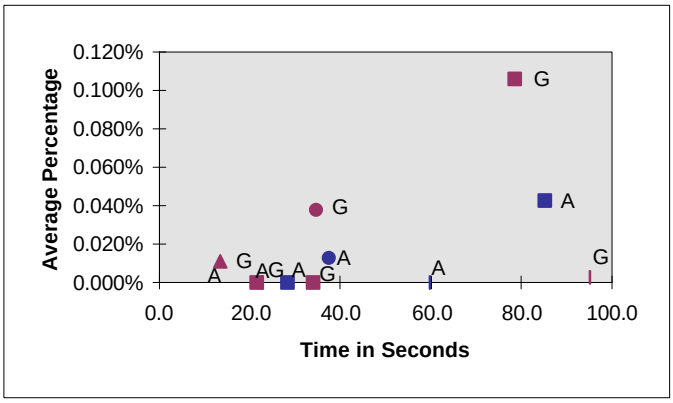

Figure 4: Average percentage deviation from optimal versus time of the ASH+LS+TS (A) and GRAH+LS+TS (G) heuristics for gap7 to gap12 instances

The explanation of the difference between the ant system and the greedy randomized adaptive heuristics is related the quality of the solution obtained by these greedy heuristics. We have observed that the solutions obtained by GRAH are very different and do not follow a patron. However, for the ASH the solutions obtained in the first iterations are worst or of equal value as the ones obtained by the GRAH, but as the search continues and the ant system heuristic is able to obtain good solutions, which leads to less running time by the local search method in the second step of the general framework. The behavior is explained by the way the ASH is designed, since good solutions seen in previous iterations are taking in account when defining the probability function for the greedy heuristic.

To exemplify the behavior of the greedy heuristics we present in Figure 5, the value of the penalty function for the initial solution obtained by the GRAH and the ASH for the instance gap9-2, using the LS+TS as the second step.

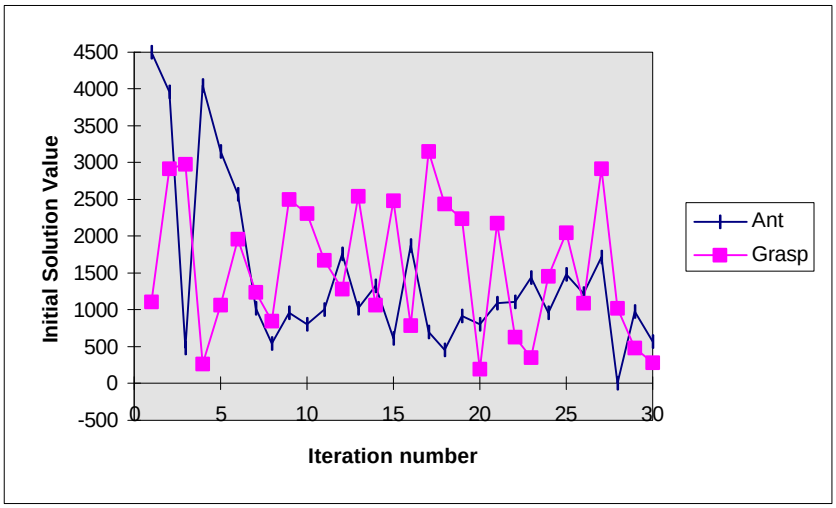

Figure 5: Penalty function value for the initial solution obtained by the Greedy Randomized Adaptive Heuristic and the Ant System Heuristic for the instance gap9-2.

5.3 Computational Results

Finally, in this last section, we present in Table 5 the performance of our best methods for all the test problems and compare them with other methods proposed to solve the GAP, the metaheuristics by Osman[5] and Chu and Beasley[3]. All the results for these methods were taking from this last work. We can observe that in average the ASH+LS+TS performed better than other approaches for these instances. This method obtains optimal solution in all runs for all test problems in gap12, meanwhile no other previous proposed method was able to it. But, our main objective here is not to declare a winner method but to understand their differences in solving different test problems. And, from the results obtained, the adaptive search heuristics proposed can obtain better or equal results for the GAP as other methods in the literature, in small running times.

Table 5: Average percentage deviation from optimal of more recent heuristics

prob.	na*nt	TS6	TS1	Ga_a	Ga_b	ASH+LS+TS	GRAH+LS+TS
gap7	8*40	0.02%	0.00%	0.08%	0.00%	0.00%	0.00%
gap8	8*48	0.14%	0.09%	0.33%	0.05%	0.04%	0.11%
gap9	10*30	0.06%	0.06%	0.17%	0.00%	0.00%	0.01%
gap10	10*40	0.15%	0.08%	0.27%	0.40%	0.01%	0.04%
gap11	10*50	0.02%	0.02%	0.20%	0.00%	0.00%	0.00%
gap12	10*60	0.07%	0.04%	0.17%	0.01%	0.00%	0.00%
Average		**0.08%**	**0.05%**	**0.20%**	**0.08%**	**0.01%**	**0.03%**

6. Conclusions

The main contribution of this work is the application of adaptive search heuristics to the generalized assignment problem, based on GRASP and Ant System methodology. The general framework approach has also some innovated aspects like the combination of the Ant Systems and GRASP with Tabu Search techniques, and the use of ejection chain neighborhoods.

Our computational testing showed that the hybrid approach based of ideas from the ant systems and the GRASP combined with tabu search leads to good results within reasonable times, and outperform the simple MMAS or GRASP. Also, the ejection chain neighborhood and the restricted candidate list strategy play an important role in driving the search to good solutions. We can also conclude that ant system based heuristics presented outperform the greedy randomized adaptive heuristics, in terms of solution quality and total running time. The results compare favorably with existing methods, both in terms of time and quality of the solution.

Further developments of this work are related to the application of the adaptive search methods to extensions of the GAP, a Resource Assignment Problem and a Pure Integer Capacitated Plant Location. Also, more work is being done for solving the more difficult problems using a sophisticated tabu search and diversification strategies, and an ant system with more ants. Moreover, as future research, we plan to apply the adaptive search heuristics based on the general framework to develop solution methods for other combinatorial optimization problems.

References

[1] Martello S and Toth P (1990). Knapsack Problems: Algorithms and Computer Implementations, Wiley: New York.

[22] Cattrysse DG and Van Wassenhove LN (1992). A survey of algorithms for the generalized assignment problem, *Eur J of Opl Res* **60:** 260-272.

[3] Chu PC and Beasley JE (1997). A genetic algorithm for the generalised assignment problem, *Comp Opns Res* **24:** 17-23.

[4] Fisher M, Jaikumar R and Van Wassenhove L (1986). A multiplier adjustment method for the generalized assignment problem, *Mgmt Sci* **32:** 1095-1103.

[5] Osman IH (1995). Heuristics for the generalized assignment problem: simulated annealing and tabu search approaches, *OR Spektrum* **17:** 211-225.

[6] Ross GT and Soland PM (1975). A branch and bound based algorithm for the generalized assignment problem, *Math Prog* **8:** 91-103.

[7] Guignard M and Rosenwein M (1989). An improved dual-based algorithm to knapsack problem, *Eur J Opnl Res* **27:** 313-323.

[8] Karabakal N, Bean JC and Lohmann JR (1992). A steepest descent multiplier adjustment method for the generalized assignment problem. Report 92-11, University of Michigan, Ann Arbor, MI.

[9] Savelsbergh M (1997). A Branch-And-Cut Algorithm for the Generalized Assignment Problem, *Opns Res* **45**: 6, 831- 841.

[10] Amini MM and Racer M (1994). A rigorous comparison of alternative solution methods for the generalized assignment problem, M*gmt Sci* **40:** 868-890.

[11] Trick MA (1992). A linear relaxation heuristic for the generalized assignment problem, *Naval Res Logist* **39:** 137-152.

[12] Cattrysse DG, Salomon M and Van Wassenhove LN (1994). A set partitioning heuristic for the generalized assignment problem, *Eur J of Opl Res* **72:** 167-174.

[13] Wilson JM (1997). A genetic algorithm for the generalised assignment problem, *J Opl Res Soc* **48:** 804-809.

[14] Laguna M, Kelly JP, González-Velarde JL and Glover F (1995). Tabu search for the multilevel generalized assignment problem, *Eur J Oper Res* **82:** 176-189.

[15] Feo TA and Resende MGC (1995). Greedy randomized adaptive search heuristic, *Journal of Global Optimization* **6:** 109-133.

[16] Stutzle T and Hoos H (1997). *Max-Min* Ant System and Local Search for Combinatorial Optimization. 2nd International Conference on Metaheuristics, Sophie-Antipolis, France.

[17] Stutzle T and Hoos H (1997). Improvements on the Ant-System: Introducing the *Max-Min* Ant System, TH Darmstadt, FB Informatik, Darmstadt, Germany.

[18] Johnson DS, Aragon CR, McGeoch LA, and Schevon C (1989). Optimization by Simulated Annealing: an experimental evaluation; part I, graph partitioning, *Opns Res* **39**:3, 865.

[19] Resende MGC, LS Pitsoulis, and Pardalos PM (1998). Fortran subroutines for computing approximate solutions of weighted MAX-SAT problems using GRASP, submitted for publication.

[20] Stutzle T (1998). An ant approach for the flow shop problem, TH Darmstadt, FB Informatik, Darmstadt, Germany.

[21] Colorni A, Dorigo M and Maniezzo V (1991). Distributed Optimization by Ant Colonies, *Proceeding of ECAL91 - European Conference on Artificial Life*: Elsevier Publishing, Paris, France, pp. 134-142.

[22] Colorni A, Dorigo M and Maniezzo V (1991). An Investigation of some Properties of an Ant Algorithm, *Proceeding of the Parallel Problem Solving from Nature Conference (PPSN 92),* Elsevier Publishing, Brussels, Belgium, pp. 509-520.

[23] Glover F (1992). Ejection Chains, Reference Structures and Alternating Path Methods for Traveling Salesman Problem, University of Colorado. *Shortened version published in Discrete Applied Mathematics,* **65:** 223:253 (1996)

[24] Glover F (1986). Future paths for integer programming and links to artificial intelligence, *Com Opns Res* **5:** 533-549.

[25] Glover F and Laguna M (1997). Tabu Search, Kluwer Academic Publishers: Norwell, Massachusetts.

[26] Barr RS, Golden BL, Kelly JP, Resende MGC and Stewart Jr WR (1995). Designing and Reporting on Computational Experiments with Heuristics Methods, *Journal of Heuristics,* **1:** 9-32.