Mathware & Soft Computing 7 (2000), 1-15

Adaptive Search Heuristics for The Generalized Assignment Problem

Helena Ramalhinho Lourenço[1], Daniel Serra [2]
[1] DEE, Universitat Pompeu Fabra, Barcelona, Spain
[2] DEE, Universitat Pompeu Fabra, Barcelona, Spain
{helena.ramalhinho, daniel.serra}@econ.upf.es

Abstract

The Generalized Assignment Problem consists of assigning a set of tasks to a set of agents at 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 agent's resource. We present the application of a MAX-MIN Ant System (MMAS) and a greedy randomized adaptive search procedure (GRASP) to the generalized assignment problem based on hybrid approaches. The MMAS heuristic can be seen as an adaptive sampling algorithm that takes into 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. Several neighborhoods are studied, including one based on ejection chains that produces good moves without increasing the computational effort. We present computational results of a comparative analysis of the two adaptive heuristics, followed by concluding remarks and ideas on future research in generalized assignment related problems.

Keywords: Generalized assignment, local search, GRASP, tabu search, ant colony optimization.

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 assigned to one and only one agent subject to capacity constraints on the agents. The GAP has applications in areas like computer and communication networks, location problems, vehicle routing and machine scheduling. Our initial interest in this problem arose from two real applications in the area of health services, resource assignment problems and pure integer capacitated plant location problems.

The aim of this paper is to present hybrid algorithms consisting of adaptive construction heuristics and subsequent application of local search to solve the GAP, and the respective computational results. The basic elements are extracted from a specific Ant Colony Optimization algorithm, *MAX-MIN* Ant System, Stützle (1997,1998a,1998b), Stützle and Hoos (1999), and GRASP algorithm, Feo and Resende (1995), Resende and Ribeiro (2001). These heuristics can be embedded in a general framework with three steps. In the first step, a solution is constructed by a randomized construction heuristic; in the second step, a local search is applied to improve this initial solution; the last step consists in updating a set of parameters which guide the construction process. 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. Second, we describe the general framework of the adaptive construction heuristics. Third, we focus on local search methods, describing the descent local search and the tabu search. Fourth, the computational experiments are described with the objective of evaluating the proposed framework and its components. Finally, we conclude with general remarks on this work and directions of future research.

2 The generalized assignment problem

The GAP consists in assigning at a minimum total cost a set of tasks to a set of agents with limited resource capacity. Each task must be assigned to one and only one agent requiring a certain amount of the agent's resource. For an extended review of this well-known problem we refer to Martello and Toth (1990), Cattrysse and Van Wassenhove (1992) and Chu and Beasley (1997). Fisher, Jaikumar and Van Wassenhove (1986) proved that the problem is *NP*-complete. Moreover, the problem of deciding if there exits a feasible solution is *NP*-hard, Sahni and Gonzalez (1976). Osman (1995) presented a survey of many real-life applications.

The Generalized Assignment Problem can be formulated as an integer program, as presented below. We use the following notation: I is the set of tasks (i=1,..., n) ; J is the set of agents (j=1,..., m) ; b_j is the resource capacity of agent j ; a_{ij} is the resource needed if task i is assigned to agent j; c_{ij} is the 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. Assume that $a_{ij} \quad b_j$ and $\sum_{i=1}^{n} a_{ij} \quad b_j$.

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

$$s.t.$$

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

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

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

Constraints (2) are the resource capacity of the agents, constraints (3) guarantee that each task is assigned to one agent, and constraints (4) are the integrality constraints.

Several exact algorithms for the GAP have been proposed by Ross and Soland (1975), Martello and Toth (1990), Fisher, Jaikumar and Van Wassenhove (1986), Guinard and Roseenwein (1989), and Karabakal, Bean, and Lohmann (1992). Recently, Savelsbergh (1997) 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 (1994) presented a variable-depth-search heuristic motivated by the work of Lin and Kernighan (1973) on the Traveling Salesman Problem. They offered a rigorous statistical analysis of the solution methods. Trick (1992) proposed an LP approximation algorithm. Cattrysse, Salomon and Van Wassenhove (1994) formulated the problem as a Set Partitioning Problem and proposed a heuristic based on column generation techniques. Osman (1995) presented a comparison of the algorithms based on tabu search and simulated annealing techniques. Wilson (1997a) presented a genetic algorithm to restore feasibility to a set of near-optimal solutions, and then to improve the best solution found by local search. Wilson also presented a simple dual algorithm for the GAP, Wilson (1997b). Chu and Beasley (1997) also presented a genetic algorithm for the GAP that tries to improve feasibility and optimality simultaneously. Laguna, Kelly, González-Velarde, and Glover (1995) proposed a tabu search algorithm based on ejection chains to an extension of the GAP, the multilevel generalized assignment problem. A variable depth search algorithm has been recently presented by Yagiura, Yamaguchi, and Ibaraki (1999) and Yagiura, Yamaguchi, and Ibaraki (1998). Diaz and Fernadez (2001) proposed a tabu search for the problem.

3 Adaptive search heuristic

This section presents the general framework and the main aspects of the adaptive heuristics proposed to solve the GAP. These adaptive heuristics are based on two metaheuristic approaches to solve combinatorial optimization problems: the Greedy Randomized Adaptive Search Procedure (GRASP), Feo and Resende (1995), Resende and Ribeiro (2001), and the *Max-Min* Ant System (MMAS), Stützle (1997,1998a,1998b), Stützle and Hoos (1999), Stützle and Hoos (2000), a specific algorithm falling into the framework of the Ant Colony Optimization metaheuristic, Dorigo and Di Caro (1999). 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 innovative aspects.

General Framework
The adaptive heuristics proposed to solve the GAP can be described in a general framework comprising three steps, that are repeated iteratively until some stopping criteria is met. In step 1 a solution is generated using a randomized construction heuristic; in step 2, a local search is applied to the solution constructed in step 1, and finally in step 3, if there exist parameters, they are updated following a certain criteria.

The objective of the above method is to put together in the same framework recently proposed metaheuristics which consist in, first, generating a starting solution for a local search using some randomized adaptive construction heuristic, then possibly updating some parameters based on the incumbent solution and continuing this process until some stopping criterion is met. Randomized adaptive construction heuristics are used because, with a simple construction heuristics, only one initial solution can be generated and randomization of the construction heuristic allows more different initial solutions. The same objective could, in principle, be obtained by restarting a local search from random initial solutions; yet, computational experience on a wide variety of combinatorial optimization problems has shown that, using random solutions do not result in very good performance. It is also outperformed by randomized construction heuristics (RCH) which take into account the contribution to the objective function value. The RCH guides their solution construction using information recently seen in good solutions, Glover (2000).

Other metaheuristics like Iterated Local Search (ILS), Lourenço, Martin and Stützle (2001), Scatter Search, Glover(1998), and Memetic Algorithms (MA), Moscato (1999) fit into the above framework. They generate new initial solutions for a subsequent improvement by local search. Yet, the main differences from GRASP and ACO algorithms are as follows: in the case of the ILS, the initial solutions are generated by modifying a current solution through a kick move; In the case of Memetic algorithms and

Scatter Search, the initial solutions are obtained by manipulating a population of previously seen solutions.

In particular, the two approaches considered in the present work, are the greedy randomized adaptive search procedure (GRASP) and a specific version of the Ant Colony Optimization Algorithms, known as *MAX-MIN* Ant Systems. Both of these methods will be discussed in detail in the next sections.

The first step of the proposed framework is based on a greedy heuristics, which constructs a solution as follows: at each iteration, an unassigned task is chosen following a priority function; next, an agent is chosen among the agents that can perform this task, and the task is assigned to it. 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 an Ant System Heuristic (ASH). The main difference between the basic greedy heuristic and the two heuristics, GRAH and ASH, is the method of selecting the agent to whom the previously chosen task is assigned. For the basic greedy heuristic, the choice is deterministic and based on the cost function. For the GRAH, the choice is a probabilistic bias according to an adaptive priority function, which does not depend on the solutions seen in previous iterations of the general framework but only on the choices done in the previous iterations of the GRAH. For the ASH, the choice is also a probabilistic bias based on desirability values, which are adapted in a reinforcement style of learning during the run of the algorithm and reflect prior performance.

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

We will admit unfeasible solutions with respect to capacity constraints, i.e. the total resource required by the tasks assigned to some agents may exceed their capacity. Unfeasible solutions will be penalized into the objective function. The main reason for admitting unfeasible solutions is that, for some solutions close to the optimal one, the capacity of the agents will be almost completely taken up, therefore any neighbor obtained by interchanging or reassigning tasks will be unfeasible 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, Glover and Laguna (1997).

The penalty function is chosen as follows:
$f'(x) = \sum_{j=1}^{m}\sum_{i=1}^{n} c_{ij}x_{ij} + \sum_{j=1}^{m} \max\left\{0, \sum_{i=1}^{n} a_{ij}x_{ij} - b_j\right\}$ where $\alpha \quad 0$ is a parameter, representing the cost of using one unit of extra capacity. If a solution is not feasible, the second term will be positive and therefore the search will look for a feasible solution. The parameter can be increased during the run to penalize unfeasible solutions, and drive the search to feasible ones.

Greedy Randomized Adaptive Procedure
The first method proposed to solve the GAP is based on the greedy randomized adaptive search procedure (GRASP). GRASP is an iterative randomized sampling technique, with two phases. The first phase consists of a greedy randomized adaptive heuristic that constructs an initial solution. It is called adaptive because the greedy function takes into account previous decisions made during the construction of a solution when considering the next choice. The second phase consists of an improvement phase, which usually corresponds to a local search method. As can be seen, the GRASP fits into the general framework proposed in the previous section. GRASP has been successfully applied to many combinatorial optimization problems, Resende and Ribeiro (2001). For a list of several GRASP applications see the web site of Resende:
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 $S_j \quad j \quad 1,\ldots,m$. ($S_j$ is the set of tasks assigned to agent *j*)
2. Construct the restricted candidate list (RCL) of agents for each task, L_i, such that $L_i \quad j: c_{ij} \quad c_{\max}$, $c_{\max}$ is a parameter that limits the dimension of the RCL (if $c_{\max} = \max_{i,j} c_{ij}$ all agents will be included).
3. Consider any order of the tasks, for example $i \quad 1,...,n$. Let $i \quad 1$.
4. While (not all tasks have been assigned) repeat:
   4.1. Choose randomly an agent *j** from L_i. Each agent in the list L_i has the following probability of being chosen: $p_{ij} \quad \frac{b_j / a_{ij}}{\sum_{l \in L_i} b_l / a_{il}}, j \quad L_i$, which depends on the resource of agent *j* and the resource needed by task *i*.

4.2. If there is available capacity, task i is assigned to agent $j^*: S_{j^*} \quad S_{j^*} \quad i$. Otherwise, the task is assigned to the first agent with spare capacity. If all agents are fully occupied, the assignment is random.

4.3. Let $i=i+1$ and if $\sum_{i \in S_{j^*}} a_{ij^*} > b_{j^*}$ remove j^* from any list. Repeat step 4 until all tasks are assigned. (Note that the capacity constraint can be violated).

5. Let $x_{ij} \quad 1 \text{ if } i \quad S_j ; x_{ij} \quad 0, \text{otherwise.}$ Calculate the value of the penalty function for the solution, $f'(x)$.

In this first step of the framework, we are looking for a feasible solution. Therefore, we are interested in assigning a task to an agent if this task uses a small amount of the agent's resource, i.e. a_{ij} / b_j is small. For each task, we order the agents in decreasing order of p_{ij}. 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 spare capacity. If all agents are fully occupied, an agent is randomly selected with uniform probability and the task is assigned to this agent. Note that the solution obtained can be unfeasible with respect to the capacity constraints. And also, that the construction heuristics does not take into considerations previously seen solutions during the run of the general framework, only the choices done previously within the greedy heuristic.

The Ant Colony Optimization

The GRASP procedure described in the previous section is a multi-start local search, but, instead of considering a random initial solution, a greedy randomized heuristic is used to try to find better initial solutions than random ones. A different way of generating initial solutions is followed by the Ant Colony Optimization (ACO) paradigm, Colorni, Dorigo and Maniezzo (1991a,1991b), Dorigo, Maniezzo, and Colorni (1996), Dorigo and Di Caro (1999). ACO is a cooperative algorithm inspired by the foraging behavior of real ants. Ants lay down in some quantity of an aromatic substance, known as pheromone, on their way to food and on their way back to the nest. Ants choose to follow a pheromone trail with a probability proportional to the pheromone trail intensity. Research on real ants has shown that such a trail following behavior allows the ants to identify shortest path between a food source and their nest, Goss, Aron, Deneubourg, and Pasteels (1989).

ACO is inspired by this behavior and tries to adapt it to the solution of combinatorial optimization problems. The ACO approach associates pheromone trails to features of the solutions of a combinatorial problem, and can be seen as a kind of adaptive memory of the previous solutions. Solutions are iteratively constructed in a randomized fashion biased by the pheromone trails left by the previous ants and possible available

information based on the problem data. In the specific ACO algorithms we are concerned with the way the pheromone trails are updated after the construction of a solution, by ensuring that the best features will have a more intensive pheromone. The first ACO algorithm was an Ant System which was applied to the TSP, Colorni, Dorigo and Maniezzo (1991a,1991b), Dorigo, Maniezzo, and Colorni (1996), Dorigo and Gambardella (1997). Subsequently several improvements of Ant System have been proposed and currently one of the best performing ACO algorithms is MAX-MIN Ant System, Stützle (1997). For more information, see the web page: http://iridia.ulb.ac.be/dorigo/ACO/ACO.html.

The *MAX-MIN* Ant system (*MMAS*) is an adaptive sampling algorithm that takes into consideration the experience gathered in earlier iterations of the algorithm. Moreover, combining MMAS with local search, Stützle and Hoos (1999), Stützle (1997, 1998a) were able to find very good solutions to the Traveling Salesman Problem, the Quadratic Assignment Problem and the Flow-Shop Scheduling Problem. Next, we will propose a MMAS approach with an additional Local Search for the GAP, which can be seen as an improvement over the Ant Colony Optimization and is currently the best performance variants of the ACO algorithms for a wide variety of combinatorial optimization problems, Stützle (1998a).

The pheromone trails, $_{ij}$, for the GAP represent the desirability of assigning a task *i* to an agent *j*. Initially, let τ_{ij} $\frac{1}{c_{ij}}$ (for GAP with a maximization objective, we set τ_{ij} c_{ij}). The cheaper the assignment of the task *i* to agent *j* is, the more desired is the assignment.

First of all, let us explain how to construct a solution using only one ant, as used in the MMAS suggested by Stützle to the flow-shop scheduling problem. Usually the MMAS are population-based heuristics but in extreme cases can be applied using only one single ant. As suggested by Stützle and Hoos (1999), the selection rule for the assignment of a task to an agent can be described as follows: with some probability p_0 the best choice is made, otherwise the assignment is done according a probability function. We designated this heuristic by Ant System Heuristic (**ASH**). 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 $_{ij}$. A task *i* is assigned to a particular agent *j* in the following way:

4.1.1. With probability p_0 , choose the agent j^* with maximal value of $_{ij}$.

4.1.2 With probability $1 - p_0$, choose the agent j^* according to the following probability distribution: $p_{ij} = \begin{cases} \frac{\tau_{ij}}{\sum_{l \in L_i} \tau_{il}} & \text{if } j \in L_i \\ 0 & \text{otherwise} \end{cases}$.

This rule was proposed by Dorigo and Gambardella (1997). The assignment obtained by the ASH 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 using the current local optimal solution in the following way:

$$\tau_{ij}^{new} = \rho \tau_{ij}^{old} + \Delta \tau_{ij},$$

where $\rho, 0 \leq \rho \leq 1$, is the persistence of the trail, i.e. $1 - \rho$ represents the evaporation. The amount of pheromone deposited by the current local optimal solution is:

$$\Delta \tau_{ij} = \begin{cases} \tau_{max} Q & \text{if task } i \text{ is assigned to agent } j \text{ in the solution} \\ 0, & \text{otherwise} \end{cases}$$

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

The values of parameter Q were set by preliminary tests following the recommendations of Stützle and Hoos (1999), and they proved to be robust. Note that, if the solution is feasible, the amount of τ_{ij} added is larger, trying to give a stronger bias to feasible assignments. Moreover, $\tau_{min} \leq \tau_{ij} \leq \tau_{max}$, $\forall i, j$, so these limits must be imposed if the updated pheromone falls outside the above interval. Also, the amount of pheromone deposit does not depend on the solution's objective function, which is different from others ACO applications, Dorigo, Di Caro (1999), but we have found this update rule very effective. Next, we present the local search methods for the second step of the general framework proposed in section 3.

4 Local search methods

In order to derive a local search method, it is necessary to define a neighborhood structure, that associates a set of solutions *N(x)* with each solution *x*. The neighborhood is usually obtained by specific modifications on *x*, called moves.

The local search starts with an initial solution and searches the neighborhood for the solution with the lowest cost. Then, this neighbor solution would replace the current solution 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.

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 task is reassigned to new agents. The shift neighborhood is a special case of the λ-generation mechanism proposed by Osman (1995). The ejection chain neighborhood is based on the work of Laguna, Kelly, González-Velarde, and Glover (1995). Other complex neighborhoods with very good results have been proposed for this problem, we refer to Yagiura, Yamaguchi, and Ibaraki (1999) and Yagiura, Yamaguchi, and Ibaraki (1998).

The move in the shift neighborhood consists in removing a task from one agent and assigning it to another agent. The size of the neighborhood is *n(m-1)*. Note that we should start by considering removing tasks from overloaded agents. The task can be reassigned to an agent with available capacity. The shift neighborhood can be obtained by the following procedure, where *flag* indicates if a neighbor solution with better value of the penalty function was found, (*flag=true*) or not (*flag=false*) (*flag* is initially set to false in the local search heuristic, see next page).

Neighborhood (*x, flag*)

1. Arrange the agents in decreasing order by the amount of capacity over the maximum allowable. Let *j=1*.
2. Consider any order of the tasks assigned to agent *j*. Let *i=1* (the first task assigned to *j*).
3. Remove *i* from the set of tasks assigned to *j*, $S_j \quad S_j \quad i$.
4. Assign *i* to another, not yet considered, agent starting with the last agent in the list obtained in step 1, (neighbor *x'* of *x*).
5. Calculate the value of the penalty function for *x', f'(x')*. If *f'(x')<=f'(x)*, let *x=x', flag=true* and stop.
6. Let *i = i +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.

In preliminary tests, this neighborhood was able to obtain feasible solutions when starting from an unfeasible one. However, the objective function values of these feasible solutions were not very good. This led us to define a more complex neighborhood, which we present next.

The ejection chain neighborhood is a variable depth procedure, which moves more than one task from the current agent to a new agent. Ejection chains were introduced by Glover (1992), and have been applied to several problems, including an extension of the GAP, Laguna, Kelly, González-Velarde, and Glover (1995). This second neighborhood

structure is more complex than the shift neighborhood, but leads to a more powerful and efficient search without increasing significantly the computation 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 (j), then assign task i to a different agent (w) as in the shift neighborhood.

Move B: Remove a task i from an agent (j), then assign task i to a different agent (w). Then, remove a task k from agent w and assign task k to another agent (different from w, but it can be agent j).

The ejection chain neighborhood of a solution can be obtained in a similar way as the shift neighborhood, but a type B move is only applied if the type A move was unsuccessful. Note also that the number of neighbors is of order $O(n^2m^2)$. As described a move in the ejection chain neighborhood is like doing two shift moves at once, one move after another one, hence, it is significantly larger than the shift neighborhood.

Descent Local Search

We designed the descent local search method with a first improvement strategy. This means that, the first cost improvement neighbor solution found becomes the new current solution. This method stops at a locally optimal solution with respect to the chosen neighborhood structure. The tabu search method presented in the next section has a different acceptance strategy from the local search and other features designed to avoid being trapped at a bad local optimum.

The main steps of the first-descent local search are:

1. Obtain an initial solution x (for example, using the GRAH). Let *flag=false*;
2. *Neighborhood*(*x*,*flag*);
3. If *flag=false*, stop (a local optimum was found), otherwise repeat step 2.

If the heuristic finishes with an unfeasible solution, apply a local search with the following neighborhood: interchange tasks between agents considering only moves that reduce the extra used capacity. After a feasible solution is obtained, we can apply the same 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}, $\sum_{s=1}^{n} a_{sj}x_{sj}$ a_{ij} a_{kj} b_j and $\sum_{s=1}^{n} a_{sl}x_{sl}$ a_{kl} a_{il} b_l, then let x_{il} x_{kj} 1 and x_{ij} x_{kl} 0.

We are now in a position to present the GRASP method:

1. While a stopping criterion is not satisfied:
   1.1. Construct a solution (*x*) using the Greedy Randomized Adaptive Heuristic. In the first iteration initialize the best solution, x_b, as *x*.
   1.2. Apply descent 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 above is the *MAX-MIN* Ant System procedure 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 *x* using the Ant System Heuristic. In the first iteration initialize the best solution, x_b, as *x*.
   2.2. Apply descent local search(*x*).

2.3. Update the pheromone trails using the current solution x.
2.4. If x is feasible, and $f'(x) \quad f'(x_b)$ let $x_b \quad x$.

3. Return the best solution found, x_b.

In both methods, the stopping criterion applied consists of 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 construction heuristic. 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, which can be considered as an intensification strategy.

Tabu Search

Tabu search was originally proposed by Glover (1986), and since then it has been subject to extensive studies and applied to several optimization problems with great success. Tabu search can be described as an intelligent search that uses memory to drive the search out of locally optimal solutions and to find good results. For a survey of tabu search see Glover and Laguna (1997).

Our reason for applying tabu search to the GAP was the excellent results obtained and by Osman (1995) and Laguna, Kelly, González-Velarde, and Glover (1995) to GAP and a multilevel generalized assignment problem, respectively. They also presented computational results for the GAP and were always able to obtain the optimal solution for test problems with 5 agents and 25 task in very short time.

The basic ingredients of the applied tabu search are: the neighborhood, the tabu list and its size, the aspiration criteria and the stopping criteria. We will now describe these aspects in detail for the GAP.

The tabu search can be briefly described as follows:

1. Generate an 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.

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 spending 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 occurs. Let x be the current solution and x' be a neighbor solution of x:

- A task u is considered for moving from an 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_{lp} ;
- A task u is considered for moving from an agent p to an agent k if $\sum_{i=1}^{n} a_{ip}x_{ip} > b_p$ and $\sum_{i=1}^{n} a_{ik}x'_{ik} \le b_k$, where x'_{ik} x_{ik} i $u, p;$ x'_{up} $0, x'_{uk}$ 1 .
- A pair of tasks u, l are considered for moving if $\sum_{i=1}^{n} a_{ip}x_{ip} > b_p$, $\sum_{i=1}^{n} a_{ik}x'_{ik} \le b_k$ and $\sum_{i=1}^{n} a_{iq}x'_{iq} \le b_q$, where x'_{ij} x_{ij} i $u, l;$ j $k, p, q;$ x'_{up} $0, x'_{uk}$ $1, x'_{lk}$ $0, x'_{lq}$ 1 .

A tabu attribute 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 this task is reassigned to agent k, then for a certain number of iterations it is forbidden to assign 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 criterion is considered which overrules the tabu status of a move, if it leads to a new best solution. Finally, the tabu search stops after a maximum number of iterations.

As we have done for the descent 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 simple descent local search method, the tabu search is applied instead.

5 Computational experiment

In this section, we will present the computational experiments and the results obtained. We have followed the guidelines proposed by Barr, Golden, Kelly, Resende, and Stewart (1995). The computational experiment was designed with two main objectives:

- To gain understanding of the behavior of the different proposed methods, based on the general framework of the adaptive construction heuristics.
- To compare the two methods proposed for the first step of the general framework: greedy randomized adaptive heuristic versus the ant system heuristic.

We would like to focus that these objectives are the most important ones, since with the computational experiment we would like to contribute to the understanding of the algorithms that follow on the framework of the adaptive search heuristics. For example, we would like to analyze if the consideration of information from previous solutions on the construction phase has an impact on the results, i.e. compare the GRAH with the ASH methods. Another questions that we would like to study are: the impact of considering or not a complete neighborhood in the local search phase, the benefits of using the tabu search method versus a local search; and the effect of using unfeasible solutions. We believe that by analyzing these questions we contribute to the advance of the knowledge of the field of heuristics in general, and adaptive search heuristics in particular by giving guidelines on the development of algorithms to real-life applications based on the GAP model and other combinatorial optimization methods.
A secondary objective is to compare the best methods described in this work with other techniques and methodology proposed to solve the GAP, but the emphasis of the paper is not to perform a competition on algorithms, since as mentioned by Hooker (1995) "competitive testing tells us which algorithms are faster but not why". As mentioned before, here, we try to ask which components make the algorithms faster and better (therefore the "why question" that Hooker talks about).

All described methods were coded in Fortran, and were tested on a set of problems ranging from 5 agents/15 jobs to 10 agents/60 jobs. The test problems are available from the OR library (http://www.ms.ic.ac.uk/info.html) and have also been used by other authors in their computational experiments, Osman (1995), Cattrysse, Salomon and Van Wassenhove (1994), Chu and Beasley (1997). The set of test problems can be divided in two groups: easy (gap 1 to gap 6) and difficult (gap7 to gap12). This set of problems is in the maximization form of GAP, so we have converted them into minimization form. All numerical tests were carried on a PC-Pentium II with 166 MHz and 16MB RAM.

Recently, a set of problems with a larger number of tasks and agents has been published. However, in our preliminary testing, the answer of the questions posed before did not change by the use of this larger size set of test problems, and the testing would be more limited because of the larger running times. We believe the use of this larger set would be more adequate for a competition type paper.

The performance measures considered are:

- Solution quality measured as the percentage deviation from the optimal solution.
- Computation time: total running time and the time to find the best solution.

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

- Problem specific: number of agents (m); number of tasks (n); 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 (descent local search and tabu search).
- Stopping criteria: number of total iterations (NTI) and the number of iterations of the tabu search (NITB).
- Other parameters: size of the RCL, size of the tabu list (STL), ant system parameters ($_{\min}$, $_{\max}$, 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. The following parameters values for *MAX-MIN* ant system were found to give good performance in preliminary runs: $_{\min} = 0.1 \times \min_{i,j} _{ij}$ and $_{\max} = n \times \max_{i,j} _{ij}$, ρ 0.75 and p_0 $\frac{n-m}{n}$ 0.8 . We have followed the guidelines of Stützle and Hoos (1999) to obtain the values for these parameters. These values are quite robust, and small changes did not affect the final results.

Comparison between different approaches

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

MMAS : ant system heuristic and descent local search with ejection chain neighborhood.

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

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

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

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

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

ASH+LS+CTS: ant system heuristic, descent 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 descent local search method with shift neighborhood. Most of the solutions obtained in the first step are unfeasible, and the descent local search with shift neighborhood usually finds a feasible one in a short time. Therefore, if a local search method is applied before the tabu search, this last one will start from a better solution. The comparison between the TS alone and the LS+TS will permit us to evaluate the performance of the TS with respect to the initial solution of the tabu method. With the last method, ASH+LS+CTS, we will try to analyze the effect of using or not a restricted candidate list.

In this experiment, the following factors are prefixed: NTI =30, NITB=200, α=50, STL=10. These values were set in preliminary tests, and they were set to control the running time. Also, since all the above methods performed very well on 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 from optimal of 5 runs for each of the heuristics proposed. First of all, we observe that the best results were obtained by the ASH+TS, GRAH+TS, ASH+LS+TS and GRAH+LS+TS, i.e. when the tabu search was used in the second step of the general framework. Also, the combination of the LS and TS improves the results, since the tabu search starts with a better solution and finds rapidly a good locally optimal solution. MMAS and GRASP obtained the worst results and they often got stuck at a bad locally optimal solution. Therefore, using a tabu approach enabled to continue the search for good solutions. When a tabu search considered the complete neighborhood, ASH+LS+CTS, the quality of the solution did not improve. So, the use of restricted candidate lists plays an important role in the search, and helps to find good solutions in significantly less time, as can be observed at the computational times given in Tables 3 and 4. It can also be seen that the proposed heuristic performs very well, finding the optimal solution in many instances. For those in which the heuristics failed to reach the optimal one, the solutions obtained are very close to optimality. In Table 2 we present the average solution quality for each set of the test problems. It will be seen 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.

In Table 3 and 4 we present the average total CPU-time and the average time to find the best solution for the 6 test problems. For all heuristics, the average and total CPU-time

increases with the ratio $\frac{n}{m}$, and also with the number of tasks. 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 computation time to find the best solution is significantly lower than the total running time, and again the ant system heuristic finds the best solution faster. However, there is a small difference between the ASH+TS, ASH+LS+TS and GRAH+LS+TS with respect to the running times to find the best solution. The explanation for this behavior is that the tabu search with the ejection chain neighborhood efficiently finds good solutions.

To better understand the behavior of the various heuristics 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 ones dominate the remaining ones. If we had to choose only one, our choice would had been the ASH+LS+TS because it obtains the best solution within a reasonable computational time.

Comparison between the adaptive approaches

A second issue that we would like to answer is related to 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 so diversify the search for a good solution. The other approach, based on the ant system, uses information on the good solutions visited in previous iterations to construct a solution, which also follows a greedy approach. We wanted to see if there is any difference between these two approaches for the GAP. Therefore, all the factors were 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. One can see 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.

The explanation of the difference between the ant system and the greedy randomized adaptive heuristics is 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 pattern. However, for the ASH the solutions obtained in the first iterations are worse or

of the same value as those obtained by the GRAH. But, as the search continues, the ant system heuristic is able to obtain good solutions, which means 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 taken into 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.

Computational Results

Finally, in this last section, we show the performance of our best methods for all the test problems in Table 5 and compare them with other methods proposed to solve the GAP, the metaheuristics by Osman (1995), TS6 and TS1, and Chu and Beasley (1997), Ga_a and Ga_b. All the results for these methods were taken from this last work. Chu and Beasley (1997) run their experiment on a Silicon Graphics Indigo (R400, 100 MHz). The fair comparison between methods is quite difficult since it depends on coding skill, machine type and speed, parameter setting, etc. The experiment was designed to evaluate the contribution to good results of components of the general framework, and not to have a comparison between algorithms. Table 5 is presented here only to give an idea of the performance of the methods proposed with respect to which has been done in the area.

We can observe that on average the ASH+LS+TS performed better than other approaches for these instances. This method obtains the optimal solution in all runs for all test problems in gap12, which no other previously proposed method was able to match. The average running time of the genetic algorithm approach for these problems was 300 CPU seconds, Chu and Beasley (1997), and for the ASH+LS+TS was less than 150 seconds. But our main objective here is not to declare a winning method but to understand their differences in solving different test problems. From the results obtained, the adaptive search heuristics proposed can obtain better or equal results for the GAP than other methods in the literature and in shorter running times.

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 Colony Optimization. The general framework has also some innovative aspects like the combination of the *MAX-MIN* Ant Systems (MMAS) and GRASP with Tabu Search techniques, and the use of ejection chain neighborhoods.

Our computational testing showed that the hybrid approach, based on ideas from MMAS and GRASP, combined with tabu search leads to good results within reasonable times, and outperforms MMAS or GRASP alone. From the results, we can conclude that the success of ASH+LS+TS relies on the combination of the tabu search and the ant system heuristics. The fact that the ASH takes into account information of previous iterations of the general algorithms is one of the main aspects to the success of the ASH+LS+TS. 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 the ant system heuristic presented outperforms the greedy randomized adaptive heuristics both in

terms of solution quality and total running time. The results compare favorably with existing methods, both in terms of time taken 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. More work is also being done in solving 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.

Acknowledgments- The research was funded by Ministerio de Ciencia y Tecnología, Spain (BEC2000-1027). The authors are grateful to Thomas Stützle and the anonymous referees whose comments have improved this paper. Also, the authors thank the Fundación BBVA-CRES for the grant award given to this work.

References

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

[2] Barr, R.S., Golden, B.L., Kelly, J.P., Resende, M.G.C. and Stewart, Jr. W,R. (1995). Designing and Reporting on Computational Experiments with Heuristics Methods, J Heuristics, 1: 9-32.

[3] Cattrysse, D.G. and Van Wassenhove, L.N. (1992). A survey of algorithms for the generalized assignment problem, Eur J of Opl Res 60: 260-272.

[4] Cattrysse, D.G., Salomon, M. and Van Wassenhove, L.N. (1994). A set partitioning heuristic for the generalized assignment problem, Eur J of Opl Res 72: 167-174.

[5] Chu, P.C. and Beasley, J.E. (1997). A genetic algorithm for the generalised assignment problem, Comp Opns Res 24: 17-23.

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

[7] Colorni, A., Dorigo, M. and Maniezzo, V. (1991b). The Ant System: Optimization by a Colony of Cooperating Agents, IEEE Transactions on Systems, Man, and Cybernetics -Part B, 26, 1, 29-41.
[8] Diaz, J.A. and Fernadez, E. (2001). A tabu search heuristic for the generalized assigmnet problem. Eur J of Opl Res 132:1 22-38.
[9] Dorigo, M. and Di Caro, G. (1999). The ant colony optimization meta-heuristic. In D. Corne, M. Dorigo, and F. Glover (eds), New Ideas in Optimization, McGraw-Hill.
[10] Dorigo, M. and Gambardella (1997). Ant Colony System: a cooperative learning approach to the traveling salesman problem. IEEE Transactions on Evolutionary Computation 1: 53-66.
[11] Dorigo, M., Maniezzo, V. and Colorni, A. (1996), The ant system: Optimization by a colony of cooperating agents, IEEE Transactions on Systems, Man, and Cybernetics – Part B, 26(1): 29-42.
[12] Feo, T.A. and Resende, M.G.C. (1995). Greedy randomized adaptive search heuristic, Journal of Global Optimization 6: 109-133.
[13] Fisher, M., Jaikumar, R. and Van Wassenhove, L. (1986). A multiplier adjustment method for the generalized assignment problem, Mgmt Sci 32: 1095-1103.
[14] Glover, F. (1986). Future paths for integer programming and links to artificial intelligence, Comp Opns Res 5: 533-549.
[15] 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).
[16] Glover, F. and Laguna, M. (1997). Tabu Search, Kluwer Academic Publishers: Norwell, Massachusetts.
[17] Glover, F. (1998). A Template for Scatter Search and Path Relinking, in Artificial Evolution, Lecture Notes in Computer Science 1363, J-K. Hao, E. Lutton, E. Ronald, M. Schoenauer and D. Snyers (Eds.), Springer-Verlag, 13-54.
[18] Glover, F. (2000). Multi-start and strategic oscillation methods – Principles to exploit adaptive memory, in Computing Tools for Modeling, Optimization and Simulation, edited by Manuel Laguna and José Luis González Velarde, Kluwer Academic Publishers, 1-38.
[19] Goss, S., Aron, S., Deneubourg, J.L., and Pasteels, J.M. (1989). Self-organized shortcuts in the Argentine Ant, Naturwissenschaften, 79: 579-581.
[20] Guignard, M. and Rosenwein, M. (1989). An improved dual-based algorithm to the knapsack problem, Eur J Opnl Res 27: 313-323.

[21] Hooker, J.N. (1995). Testing heuristics. We have it all wrong. J Heuristics, 1: 33-42.
[22] Karabakal, N., Bean, J.C. and Lohmann, J.R. (1992). A steepest descent multiplier adjustment method for the generalized assignment problem. Report 92-11, University of Michigan, Ann Arbor, MI.
[23] Laguna, M., Kelly, J.P., González-Velarde, J.L. and Glover, F. (1995). Tabu search for the multilevel generalized assignment problem, Eur J Oper Res 82: 176-189.
[24] Lin, S. and Kernighan, B.W. (1973), An efficient heuristic algorithm for the traveling salesman problem, Operations Research 21: 498-516.

[25] Lourenço, H.R., Martin, O. and Stützle, T. (2001). Iterated Local Search. To appear in State-of-the-Art Handbook of Metaheuristics, F. Glover and G. Kochenberger, eds. Kluwer Academic Publishers.
[26] Martello, S. and Toth, P. (1990). Knapsack Problems: Algorithms and Computer Implementations, Wiley: New York.
[27] Moscato, P. (1999). Memetic Algorithms: a short introduction, in New Ideas in Optimization, edited by D. Corne, F. Glover, and M. Dorigo, McGraw-Hill, 219-234.
[28] Osman, I.H. (1995). Heuristics for the generalized assignment problem: simulated annealing and tabu search approaches, OR Spektrum 17: 211-225.
[29] Resende, M.G.C. and Ribeiro, C.C. (2001). Greedy Randomized Adaptive Search Procedures. To appear in State-of-the-Art Handbook of Metaheuristics, F. Glover and G. Kochenberger, eds. Kluwer Academic Publishers.
[30] Ross, G.T. and Soland, P.M. (1975). A branch and bound based algorithm for the generalized assignment problem, Math Prog 8: 91-103.
[31] Sahni, S. and Gonzalez, T. (1976). P-Complete Approximation Problems. J ACM, 23: 555-565.
[32] Savelsbergh, M. (1997). A Branch-And-Cut Algorithm for the Generalized Assignment Problem, Opns Res 45: 6, 831- 841.
[33] Stützle, T. (1997). MAX-MIN Ant System for the Quadratic Assignment Problem, Technical Report AIDA-97-4, FG Intellektik,TU Darmstadt, Germany.
[34] Stützle, T. (1998a). Local Search Algorithms for Combinatorial Problems-Analysis, Improvements , and New Applications. PhD thesis, Departement of Computer Science, Darmstadt University of Technology, Germany.
[35] Stützle, T. (1998b). An ant approach for the flow shop problem, In Proceeding of the 6th European Congress on Intelligent Techniques & Soft Computing (EUFIT'98), 3: 1560-1564, Verlag Mainz.

[36] Stützle, T. and Hoos, H. (1999). Max-Min Ant System and Local Search for Combinatorial Optimization. In: S. Voß S. Martello, I.H. Osman and C. Roucairol (eds), Meta-Heuristics: Trends in Local Search paradigms for Optimization, Kluwer Academic Publishers, pp. 313-329.

[37] Stützle, T. and Hoos, H. (2000). Max-Min Ant System. Future Generation Computer Systems 16: 889-914.

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

[39] Yagiura, M., Yamaguchi, T. and Ibaraki, T. (1998). A variable depth search algorithm with branching search for the generalized assignment problem, Optimization Methods and Software, vol. 10, 419-441.

[40] Yagiura, M., Yamaguchi, T. and Ibaraki, T. (1999). A variable depth search algorithm for the generalized assignment problem, in: S. Voß S. Martello, I.H. Osman and C. Roucairol (eds), Meta-Heuristics: Trends in Local Search paradigms for Optimization, Kluwer Academic Publishers, 459-471.

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

[42] Wilson, J.M. (1997b). A simple dual algorithm for the generalized assignment problen, J Heuristics 2(4): 303-311.