GREEDY RANDOMIZED ADAPTIVE SEARCH PROCEDURES

LEONIDAS S. PITSOULIS AND MAURICIO G.C. RESENDE

Abstract. GRASP (greedy randomized adaptive search procedure) is a metaheuristic for combinatorial optimization. GRASP usually is implemented as a multistart procedure, where each iteration is made up of a construction phase, where a randomized greedy solution is constructed, and a local search phase which starts at the constructed solution and applies iterative improvement until a locally optimal solution is found. This chapter gives an overview of GRASP. Besides describing the basic building blocks of a GRASP, the chapter covers enhancements to the basic procedure, including reactive GRASP, hybrid GRASP, and intensification strategies.

1. Introduction

Consider a *combinatorial optimization* problem, where one is given a discrete set X of solutions and an objective function $f(x) : x \in X \to \mathbb{R}$ to be minimized and seeks a solution $x^* \in X$ such that $f(x^*) \leq f(x)$, for all $x \in X$. Problems of this type are sometimes easy to solve, i.e. they can be solved in polynomial time, but more often polynomial-time algorithms (Garey and Johnson, 1979a) to solve them are not known and one usually resorts to heuristics that are not guaranteed to find an optimal solution in polynomial time.

Local search (see Section x.xx in this handbook) is based on searching a local neighborhood for an improving solution. Given a solution $x \in X$, the elements of the *neighborhood* $N(x)$ of x are those solutions that can be obtained by applying an elementary modification (often called a *move*) to x. For example, consider $x = (0, 1, 0)$ and the *1-flip* neighborhood of a 0/1 array, where neighbors are all 0/1 arrays that differ from x by exactly one element. For this example $N(x) = \{(0, 0, 0), (1, 1, 0), (0, 1, 1)\}$. Now consider $x = (2, 1, 3)$ and $2 - swap$ neighborhood of a permutation array. We have $N(x) = \{(3, 1, 2), (2, 3, 1), (1, 2, 3)\}$. Local search starts from an initial solution $x^0 \in X$ and iteratively generates a series of improving solutions $x^1, x^2, \ldots, x^k$. At the k-th iteration, $N(x^k)$ is searched for an improving solution x^{k+1} such that $f(x^{k+1}) < f(x^k)$. If such a solution is found, it is made the current solution. Otherwise, the search ends with x^k as a local optimum.

The effectiveness of local search depends on several factors, such as theneighborhood structure, the function to be minimized, and the starting solution. A solution x is said to be in the *basin of attraction* of the global optimum if local search starting from x leads to the global optimum. Once the neighborhood and objective function are determined, different starting solutions can be used to start the local search in a *multi-start* procedure. If the starting solution is in the basin of attraction of the global optimum, local search finds the global optimum. Otherwise, a nonglobal local minimum is found. By repeatedly using a randomly generated solution as the

Date: January 18, 2001.
AT&T Labs Research Technical Report.

starting solution for local search, one can expect to eventually produce a starting solution in the basin of attraction of the global optimum.

Randomly generated solutions are of poor quality on the average. Even if a randomly generated solution is in the basin of attraction of a good quality local optimal solution, the number of moves needed to reach the local optimum can be large, even exponential in the problem size (Johnson et al., 1988). The greedy algorithm usually produces solutions of better quality than those of randomly generated solutions. A greedy algorithm constructs a solution, one element at a time. At each step in the construction, a set C of candidate elements that can be added to the solution at that step, is constructed. A *greedy function* is applied to each candidate element and the elements are ranked according to their greedy function values. A best ranked element is added to the solution. Taking this into account, the set C is updated, causing the greedy function to change. The process is repeated until $C = \emptyset$.

With randomly generated initial solutions, a multi-start method eventually will find a global optimum. However, using greedy solutions as starting points for local search in a multi-start procedure will usually lead to good, though, most often, suboptimal solutions. This is because the amount of variability in greedy solutions is small and it is less likely that a greedy starting solution will be in the basin of attraction of a global optimum. If there are no ties in the greedy function values or, if a deterministic rule is used to break ties, there is no variability and a multi-start procedure would produce the same solution in each iteration. A *semi-greedy heuristic* (Hart and Shogan, 1987; Feo and Resende, 1989) adds variability to the greedy algorithm. After the candidate elements are ranked according to their greedy function values, well ranked candidate elements are placed in a *restricted candidate list* (RCL) and an element from the RCL is selected at random and is added to the solution.

Hart and Shogan (1987) and Feo and Resende (1989) proposed two schemes to build an RCL. In the *cardinality* based scheme, an integer k is fixed and the k top ranked candidates are placed in the RCL. In the *value* based scheme, all candidate elements with greedy function values within $\alpha\%$ of the greedy value are placed in the RCL, where $\alpha \in [0, 100]$. A scheme that is less prone to scaling problem was proposed by Resende et al. (2000), where the RCL elements are those which are within $\alpha(\bar{g} - \underline{g})$ of the greedy function value, where $\bar{g} = \max\{g(c)|c \in C\}$ and $\underline{g} = \min\{g(c)|c \in C\}$. For a minimization problem, RCL $= \{c \in C \mid g(c) \leq \underline{g} + \alpha(\bar{g} - \underline{g})$. Note that if $\alpha = 0$, then the semi-greedy construction reduces to a greedy algorithm, and if $\alpha = 1$, it is random construction.

We can now describe a basic greedy randomized adaptive search procedure (GRASP) (Feo and Resende, 1989, 1995) for a minimization problem. Let x^* is the best solution found and $f^* = f(x^*)$.

1. $f^* = \infty$;
2. Repeat until a stopping criterion is satisfied:
   (a) Generate a greedy randomized solution x;
   (b) Find local optimum x_l with local search starting from x;
   (c) If $f(x_l) < f^*$ then
      (i) $f^* = f(x_l)$;
      (ii) $x^* = x_l$;

2. The restricted candidate list

The cardinality-based RCL construction mechanism has not been frequently used in the literature. Most GRASP implementations have used some type of value-based RCL construction scheme. In such a scheme, the RCL parameter α determines the level of greediness or randomness in the construction. Early implementations of GRASP used a fixed value for α, usually determined through experimentation. In some cases, however, simply changing the problem class, or the instance of a particular problem class, required a different parameter value. Mockus et al. (1997) pointed out that, excluding the case of random construction, a GRASP with a fixed RCL parameter value may not converge (asymptotically) to a global minimum because the construction mechanism may exclude all solutions in the basin of attraction of a global minimizer.

A few remedies for this have been proposed. Resende et al. (2000) proposed using a different randomly generated $\alpha \in \text{UNIF}[0,1]$ in each GRASP iteration. A mechanism for a self adjusting α was proposed by Prais and Ribeiro (2000) in a strategy they named *Reactive GRASP*. As in Resende et al. (2000), a different randomly generated α is used in each GRASP iteration. However, instead of being chosen from a uniform distribution, α is chosen from a discrete set of values $\{\alpha_1, \alpha_2, \ldots, \alpha_m\}$. The probability that α_k is selected is $p(\alpha_k)$. Reactive GRASP adaptively changes the probabilities $\{p(\alpha_1), p(\alpha_2), \ldots, p(\alpha_m)\}$ to favor values of α that produce good solutions. Let f^* be the value of the best solution found so far during a GRASP run and let a_i be the average value of the solutions found using α_i in the construction phase. Periodically (say every N_α GRASP iterations), the values $q_i = f^*/a_i$ and $p(\alpha_i) = q_i / \sum_{j=1}^{m} q_j$ are computed for $i = 1, \ldots, m$. Note that the more suitable a value of α_i is, the larger the value of q_i is, and consequently, the higher the value of $p(\alpha_i)$, making α_i more likely to be selected.

An alternative to using a RCL was proposed by Bresina (1996). In this scheme candidates are not excluded from selection during construction, but are assigned probabilities of being selected that are determined by their greedy function values. The greedy function $g(c)$ is applied to all candidate elements $c \in C$ and a bias b_r is assigned to the r-th ranked element. Several biases are proposed. In logarithmic bias, $b_r = 1/\log(r+1)$; in linear bias, $b_r = 1/r$; in polynomial(n) bias, $b_r = 1/r^n$; in exponential bias, $b_r = 1/e^r$; and in random bias, $b_r = 1$. The probability that the r-th ranked candidate element is selected is $p(r) = b_r / \sum_{j=1}^{|C|} b_j$. Binato et al. (2001) used bias function to select an element from the RCL.

Glover and Laguna (1997) referred to the observation that "good solutions at one level are likely to be found 'close to' good solutions at an adjacent level" as the *proximate optimality principle* (POP). Fleurent and Glover (1999) pointed out that POP can be interpreted in relation to GRASP. Imperfections introduced during the steps of GRASP construction can be "ironed-out" by applying local search during (and not only at the end of) construction. POP local search is not applied after each construction iteration, but rather two or three times during construction (Fleurent and Glover, 1999; Binato et al., 2001). This avoids the reduction in solution variability that would result if it is applied after each construction step.

3. Long-term memory

One possible shortcoming of the standard GRASP framework is the independence of the GRASP iterations, i.e. the fact that GRASP does not *learn* from

the history of solutions found in previous iterations. This is so because the standard GRASP discards information about any solution encountered that does not improve upon the incumbent. Besides Reactive GRASP, which can be viewed as a long-term memory strategy, other strategies that make use of information gathered throughout GRASP iterations have been proposed, either to speed up the solution process, avoiding unnecessary searches, or to intensify the search around good-quality solutions.

A *hash table* (Aho et al., 1974; Cormen et al., 1990) is a data structure for implementing dictionaries (dynamic sets with the operations of insert, delete, and search). The expected time to search an element in a hash table is $O(1)$, which makes hash tables a computationally effective data structure. A hash table is used in GRASP to keep track of all solutions used as initial solutions for local search. After construction of the greedy randomized solution, the hash table is consulted to verify if the solution is new. If it is, it is added to the hash table, and local search is applied starting from that solution. Otherwise, local search is skipped, and a new GRASP iterations is started.

Path relinking was first introduced in the context of tabu search (Glover and Laguna, 1997) as an approach to integrate intensification and diversification strategies in the search. See Glover et al. (2000) for a survey of path relinking. It explores trajectories that connect high quality solutions by starting from an *initial solution* and generating a path in the neighborhood of this solution towards another solution, called the *guiding solution*. This path is generated by selecting moves that introduce in the initial solution attributes of the guiding solution. At each step, all moves that incorporate attributes of the guiding solution are analyzed and the move that best improves (or least deteriorates) the initial solution is chosen. Path relinking in the context of GRASP was first introduced by Laguna and Martí (1999).

A refinement of GRASP with path relinking was made by Aiex et al. (2000a). A pool P of elite solutions is formed with the solutions found in the first $|P|$ GRASP iterations. After this initial phase, each solution s_g produced by the GRASP local search phase is relinked with one or more elite solutions. Given s_g and elite solution s_e, two paths are generated, one from s_g to s_e, and another from s_e to s_g. Two paths are used because they often visit different intermediate solutions. Two strategies for selecting s_e are used. The first is the one proposed by Laguna and Martí, where s_e is selected at random from the pool. The second, relinks s_g with all elite solutions in P.

Laguna and Martí update their pool by maintaining in it three best-quality solutions. Aiex et al. used an approach proposed by Fleurent and Glover (1999) for using elite solutions within the GRASP framework. The elite set is made up of $p = |P|$ elements. Let c_{best} and c_{worst} be the objective function values of the best and the worst solutions in P, respectively. Given two solutions s and t, let $\Delta(s, t)$ be a measure of dissimilarity of solutions s and t.

Solution s_{gmin} output from the path relinking procedure is a candidate for insertion into the pool and is accepted if it satisfies one of the following acceptance criteria:

1. $c_{gmin} < c_{best}$, i.e. s_{gmin} is the best solution found so far;

2. $c_{best} < c_{gmin} < c_{worst}$ and for all elite solutions $s_p \in P$, $\Delta(s_{gmin}, s_p) > \delta$, where δ is a cut-off parameter, i.e. s_{gmin} is better than the worst elite solution and differs significantly from all elite solutions.

Once accepted for insertion into P, s_{gmin} replaces the worst elite solution, which is discarded from P.

Path relinking can also be used as an intensification phase for the elite set. This is accomplished by applying path relinking to each pair of elite solutions in the pool and updating the pool if necessary. The procedure is repeated until no further change in the pool occurs. This type of intensification can be done in a post-optimization phase (using the final pool of elite solutions), or periodically during the optimization (using the current set of elite solutions).

When applying path relinking as a post-optimization step, after no further change in the elite set occurs, the local search procedure is applied to each elite solution, as the solutions produced by path relinking are not always local optima. The local optima found are candidates for insertion into the elite set. If a change in the elite set occurs, the entire post-processing step is repeated.

Another use of the information obtained from the "good" solutions is to implement a memory-based procedure to influence the construction phase by modifying the probabilities assigned to each RCL element. Fleurent and Glover (1999) introduced a memory-based scheme that uses long-term memory in multi-start heuristics such as GRASP. Their scheme maintains a set P of elite solutions to be used in the construction phase. As in the description of the elite set for path relinking, for a solution s to become elite, it must be either better than the best member of P, or better than the worst member of P, and sufficiently different from the other elite solutions. For example, one can count identical solution vector components and set a threshold for rejection. A *strongly determined variable* is one that cannot be changed without eroding the objective or changing significantly other variables. A *consistent variable* is one that receives a particular value in a large portion of the elite solution set. Let $I(c)$ be a measure of the strongly determined and consistent features of choice c, i.e. $I(c)$ becomes larger as e resembles solutions in elite set P. The intensity function $I(c)$ is used in the construction phase as follows. Recall that $g(c)$ is the greedy function. Let $E(c) = F(g(c), I(c))$ be a function of the greedy and the intensification functions. For example, $E(c) = \lambda g(c) + I(c)$. The intensification scheme biases selection from the RCL to those elements c with a high value of $E(c)$ by setting the probability of selecting c to be $p(c) = E(c)/\sum_{s \in \mathrm{RCL}} E(s)$. The function $E(c)$ can vary with time by changing the value of λ, e.g. initially λ is set to a large value and when diversification is called for, λ is decreased. A procedure for changing the value of λ is given by Fleurent and Glover. See also Binato et al. (2001) for an application of this long-term memory strategy.

4. GRASP in hybrid metaheuristics

It is easy to incorporate GRASP in a hybrid metaheuristic. Since there is freedom to choose the local search algorithm, a number of metaheuristics can be chosen for this task. See Laguna and González-Velarde (1991), Colomé and Serra (1998), Delmaire et al. (1999), and Liu et al. (2000) for examples of the use of tabu search and simulated annealing as a GRASP local search procedure. Ahuja et al. (1998) use

TABLE 1. CPU time (in seconds) and speed-up on MAX-SAT problems. Average speed-up is shown for 5, 10, and 15 processors.

	1 processor	5 processors		10 processors		15 processors	
problem	time	time	speed-up	time	speed-up	time	speed-up
jnh201	310.4	62.8	4.9	30.5	10.2	22.2	14.0
jnh202	312.2	59.8	5.2	31.2	10.0	23.4	13.3
jnh203	351.2	72.3	4.9	35.2	10.0	23.2	15.1
jnh205	327.8	63.4	5.2	32.1	10.2	22.5	14.6
jnh207	304.7	56.7	5.4	29.6	10.3	19.8	15.4
jnh208	355.2	65.6	5.4	33.2	10.7	21.0	16.9
jnh209	339.0	60.5	5.6	33.6	10.1	21.6	15.7
jnh210	318.5	57.6	5.5	32.5	9.8	20.8	15.3
jnh301	414.5	85.3	4.9	45.2	9.2	28.3	14.6
jnh302	398.7	88.6	4.5	48.2	8.3	27.0	14.7
average speed-up:			5.2		9.9		15.0

local search with large neighborhoods in a GRASP framework. Variable neighborhood search (Hansen and Mladenović, 1998) is another candidate for hybridization with GRASP.

Path relinking, which was described in the previous section, can be incorporated into GRASP to form a powerful metaheuristic (Laguna and Martí, 1999; Aiex et al., 2000a).

GRASP has also been used in conjunction with genetic algorithms as a mechanism to generate initial solutions (Ahuja et al., 2000), as well as in crossover (Lourenço et al., 1998).

5. Parallel GRASP

As with any multi-start heuristic for combinatorial optimization, a GRASP can be implemented in parallel by dividing the k independent iterations among ρ processors. Another approach is to give a target value τ of the objective function to each processor and execute the algorithm until the first processor finds a solution with value at least as good as τ, at which point all processors halt. Some care is needed to assure that no two iterations start with identical random number generator seeds (Pardalos et al., 1996). These are examples of *multiple independent walk* parallelism (Verhoeven and Aarts, 1995).

Many parallel implementations of GRASP using the above strategies have been reported in the literature, e.g. (Martins et al., 2000, 1998; Murphey et al., 1998b; Pardalos et al., 1995, 1996). In most of these papers, a common observation was made. The speedups in the measured running times were proportional to the number of processors. A typical example can be seen in Pardalos et al. (1996) where, for a PVM-based implementation of a parallel GRASP for the MAX-SAT, average speed-ups almost identical to the number of processors were measured (see Table 1).

This observation can be explained if the random variable *solution time to target* is exponentially distributed, as indicated by the following proposition (Verhoeven and Aarts, 1995).

Proposition 1. *Let $P_\rho(t)$ be the probability of not having found a given (target) solution in t time units with ρ independent processes. If $P_1(t) = e^{-t/\lambda}$ with $\lambda \in \mathbb{R}^+$, i.e. P_1 corresponds to an exponential distribution, then $P_\rho(t) = e^{-\rho t/\lambda}$.*

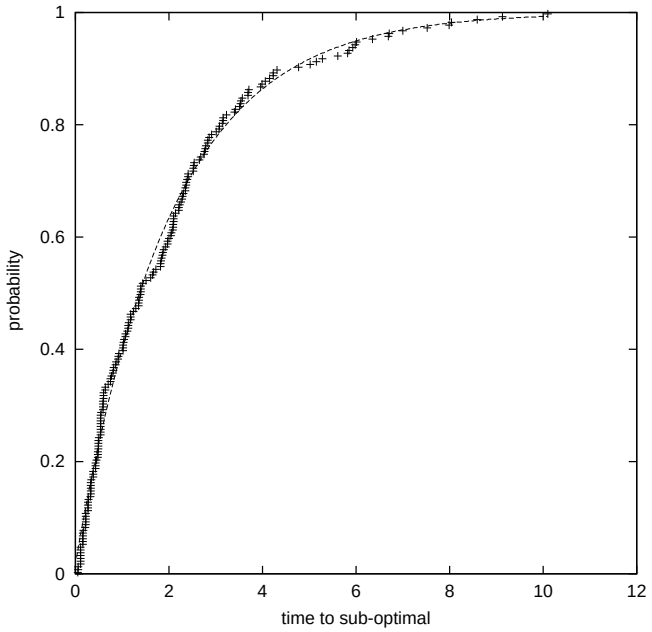

FIGURE 1. Superimposed empirical and theoretical distributions.

The above proposition follows from the definition of the exponential distribution. It implies that the probability of finding a solution of a given value in time ρt with a sequential process is equal to the probability of finding a solution at least as good as that given value in time t with ρ independent parallel processes. Hence, it is possible to achieve linear speed-up in solution time to target solution by multiple independent processes.

An analogous proposition can be stated for a two parameter (shifted) exponential distribution.

Proposition 2. *Let $P_\rho(t)$ be the probability of not having found a given (target) solution in t time units with ρ independent processes. If $P_1(t) = e^{-(t-\mu)/\lambda}$ with $\lambda \in \mathbb{R}^+$ and $\mu \in \mathbb{R}$, i.e. P_1 corresponds to a two parameter exponential distribution, then $P_\rho(t) = e^{-\rho(t-\mu)/\lambda}$.*

Analogously, this proposition follows from the definition of the two parameter exponential distribution. It implies that the probability of finding a solution of a given value in time ρt with a sequential process is equal to $1 - e^{-(\rho t-\mu)/\lambda}$ while, the probability of finding a solution at least as good as that given value in time t with ρ independent parallel processes is $1 - e^{-\rho(t-\mu)/\lambda}$. Note that if $\mu = 0$, then both probabilities are equal and correspond to the non-shifted exponential distribution. Furthermore, if $\rho\mu \ll \lambda$, then the two probabilities are approximately equal and it is possible to approximately achieve linear speed-up in solution time to target solution by multiple independent processes.

Aiex et al. (2000b) studied the empirical probability distributions of the random variable *time to target solution* in five GRASP implementations. They showed that, given a target solution value, the time it takes GRASP to find a solution at least as good as the target fits a two-parameter exponential distribution. As an example,

Figure 1 shows a plot of the empirical and estimated theoretical distributions for GRASP on an particular instance. This plot was generated using 200 independent runs of GRASP and the methodology described in Aiex et al. (2000b).

6. Applications of GRASP

Since the late 1980s GRASP has been applied to a wide range of operations research and industrial optimization problems. These include problems in scheduling, routing, logic, partitioning, location and layout, graph theory, assignment, manufacturing, transportation, telecommunications, automatic drawing, electrical power systems, and VLSI design. Festa and Resende (2001) presented an extensive annotated bibliography of the GRASP literature. In this section, we will review a small part of these applications, limiting ourselves to applications in logic, assignment, and location.

6.1. **Logic.**

6.1.1. *The satisfiability problem.* Let $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_m$ be m clauses, where each clause $\mathcal{C}_i$ contains n_i Boolean variables $x_1, x_2, \ldots, x_{n_i}$, which can take on only the values `true` or `false` (1 or 0). Each clause $\mathcal{C}_i$ contains a set of literals connected by the **or** ($\vee$) operator,

$$\mathcal{C}_i = \bigvee_{j=1}^{n_i} l_{ij},$$

where the literals $l_{ij} \in \{x_i, \bar{x}_i \mid i = 1, \ldots, n\}$, and $\bar{x}_i$ is the *negation* of x_i. A formula is a conjunction of clauses, that is, a set of clauses connected by the **and** ($\wedge$) operator,

$$\mathcal{C}_1 \wedge \mathcal{C}_2 \wedge \cdots \wedge \mathcal{C}_m.$$

In the satisfiability problem (SAT) we are given m clauses involving n boolean variables, and we are to determine whether there exists an assignment of truth values to the boolean variables, such that the formula evaluates to `true`. SAT was the first problem shown to be NP-complete (Cook, 1971; Garey and Johnson, 1979b). The satisfiability problem is a central problem in artificial intelligence, mathematical logic, and combinatorial optimization. Problems in computer vision, VLSI design, databases, automated reasoning, computer-aided design and manufacturing, involve the solution of instances of the satisfiability problem. Furthermore, SAT is the basic problem in computational complexity (Cook, 1971; Garey and Johnson, 1979b). Developing efficient exact algorithms and heuristics for satisfiability problems can lead to general approaches for solving combinatorial optimization problems. Resende and Feo (1996) presented four GRASP implementations for solving the SAT problem, with extensive computational results. In their paper they demonstrate how different GRASP implementations can be achieved by changing the greedy adaptive function of the construction phase, and the local search that GRASP performs. Their computational experiments where conducted on most of the problems from the second DIMACS Challenge benchmark problems for SAT (Johnson and Trick, 1996), and comparisons were made with GSAT (Selman et al., 1992).

6.1.2. *Inferring logical clauses from examples.* The problem of inferring logical clauses from examples is also related to the satisfiability problem. An *example* with complete data is a vector $e \in \{0,1\}^n$, while an example with incomplete data is a vector $e \in \{0,1,*\}^n$. An example can be thought of as an assignment of truth values to n boolean variables. The $*$ symbol indicates that there is no assignment (it can take the values of either true or false). Given sets of examples E^+ and E^-, the problem is to find a formula (with preferably small number of clauses) such that it will take the value of `true` for all examples in E^+, and the value of `false` for all examples in E^-. Deshpande and Triantaphyllou (1998) presented two GRASPs for solving the above problem, one for solving the problem with complete data, and the other with incomplete data. Computational experiments where conducted on problems involving 18,000 examples. Previous best results involved only up to 1,000 examples.

6.1.3. *Maximum satisfiability.* The weighted maximum satisfiability problem (MAX-SAT) is a generalization of the satisfiability problem. In the MAX-SAT, for each clause C_i we are given a weight w_i, and the problem is to find an assignment of truth values to the boolean variables $x_1, \ldots, x_n$ such that the sum of the weights of the satisfied clauses is maximized. Resende, Pitsoulis and Pardalos (Resende et al., 2000; Pardalos et al., 1996; Resende et al., 1997) presented a series of papers where GRASP is implemented for solving the MAX-SAT problem.

In Resende et al. (1997) the following mixed integer programming formulation of the MAX-SAT is used for the GRASP implementation. Let $y_j = 1$ if Boolean variable x_j is `true` and $y_j = 0$ otherwise. Furthermore, the continuous variable $z_i = 1$ if clause $\mathcal{C}_i$ is satisfied and $z_i = 0$ otherwise. Consider the mixed integer linear program

$$\max \quad F(y,z) = \sum_{i=1}^{m} w_i z_i$$

subject to

$$\sum_{j \in I_i^+} y_j + \sum_{j \in I_i^-} (1 - y_j) \geq z_i, \quad i = 1, \ldots, m,$$

$$y_j \in \{0,1\}, \quad j = 1, \ldots, n,$$

$$0 \leq z_i \leq 1, \quad i = 1, \ldots, m,$$

where I_i^+ (resp. I_i^-) denotes the set of variables appearing unnegated (resp. negated) in clause $\mathcal{C}_i$. Resende et al. (1997) described in detail the GRASP implementation for the MAX-SAT and conducted extensive computational experiments from modified SAT instances taken from the second DIMACS implementation challenge (Johnson and Trick, 1996), where the optimal solution was also computed using the CPLEX solver for comparison. The design and implementation of the actual fortran subroutines is described in Resende et al. (2000). Moreover a parallel implementation of GRASP for solving the MAX-SAT is given in Pardalos et al. (1996), where the computational results indicate average linear speedup.

6.2. **Assignment and location problems.** GRASP has been successfully applied to many combinatorial optimization problems arising in the context of assignment and location problems. Among others we mention intermodal trailer assignment (Feo and González-Velarde, 1995), the generalized assignment problem (Lourenço and Serra, 1998), satellite traffic assignment (Prais and Ribeiro, 2000), aerial photographic map mosaicing (Fernández and Martí, 1999), and location of concentrators (Resende and Ulular, 1997). In the following, examples of GRASP implementations for assignment and locations problems that have appeared in the literature are given.

6.2.1. *Quadratic assignment problem.* The quadratic assignment problem (QAP) was introduced by Koopmans and Beckmann (1957) as a mathematical model for the location of a set of indivisible economical activities. Consider the problem of allocating a set of facilities to a set of locations, with the cost being a function of the distance and flow between the facilities, plus costs associated with a facility being placed at a certain location. The objective is to assign each facility to a location such that the total cost is minimized. Specifically, we are given three $n \times n$ input matrices with real elements $F = (f_{ij})$, $D = (d_{kl})$ and $B = (b_{ik})$, where f_{ij} is the flow between the facility i and facility j, d_{kl} is the distance between location k and location l, and b_{ik} is the cost of placing facility i at location k. The Koopmans-Beckmann version of the QAP can be formulated as follows. Let n be the number of facilities and locations and denote by N the set $N = \{1, 2, \ldots, n\}$, then the QAP is formulated as

$$\min_{p \in \Pi_N} \sum_{i=1}^{n} \sum_{j=1}^{n} f_{ij} d_{p(i)p(j)} + \sum_{i=1}^{n} b_{ip(i)},$$

where Π_N is the set of all permutations $p : N \to N$. Each individual product $f_{ij} d_{p(i)p(j)}$ is the cost of assigning facility i to location $p(i)$ and facility j to location $p(j)$. In the context of facility location, matrices F and D are symmetric with zeros in the diagonal, and all the elements of the matrices are nonnegative. The QAP is one of the hardest optimization problems and, until now, no exact algorithm has been able to solve problems of size $n > 30$ in reasonable computational time. In fact, Sahni and Gonzalez (1976) have shown that the QAP is NP-hard and that even finding an approximate solution within some constant factor from the optimal solution cannot be done in polynomial time unless P = NP. There are numerous applications of the QAP in different fields, for example, hospital layout (Elshafei, 1977), ranking of archaeological data (Krarup and Pruzan, 1978), ranking of a team in a relay race (Heffley, 1977), scheduling parallel production lines (Geoffrion and Graves, 1976), and analyzing chemical reactions for organic compounds (Ugi et al., 1979).

Li et al. (1994) presented a GRASP for the QAP and reported computational results on problems from the QAP test problem set QAPLIB (Burkard et al., 1997). The authors describe the construction stage and present different local search strategies based on different neighborhood structures for the QAP. Computational results showed that GRASP finds the best known solution for almost all instances in QAPLIB, and in some cases it produces a better solution than previously reported. Resende et al. (1996) presented in detail the design, implementation, and usage of the code from (Li et al., 1994), and performed extensive computational results

where they demonstrate the effect of running time in the solution quality of GRASP for solving the QAP.

Define the *sparsity* of a matrix to be the ratio of the number of zero entries to the number of all the entries in the matrix. Given a QAP instance with input matrices F and D, define the sparsity of the instance to be the maximum sparsity of F and D. Pardalos et al. (1997) modified the GRASP for QAP presented in Li et al. (1994) to exploit sparsity. In Pardalos et al. (1997) the Fortran subroutines are described, and computational results are presented where the dense and sparse versions of GRASP for the QAP are compared in terms of solution time. The results show that, on average, the sparse version is 32% faster than the dense version without affecting the solution quality, while when sparsity is large (≥ 0.8), the sparse version can be up to 300% faster. They take advantage of the inherent parallel nature of the GRASP for the QAP, to implement a parallel version of the heuristic. Pardalos et al. (1997) presented computational results on a parallel GRASP implementation for solving large scale QAP instances, using a Kendall Square Research KSR-1 parallel computer, using up to 64 processors in parallel. The results indicate an average speedup that is almost linear with respect to the number of processors.

GRASP can be thought of as a constructive multistart procedure, that is, a procedure which performs local search each time using a different initial solution obtained from a construction procedure. An immediate observation is that GRASP is *memoryless*, or it does not use information that could be obtained from previously computed solutions for the construction of new solutions. Fleurent and Glover (1999) suggest modifications in which information from previous solutions can be incorporated into GRASP for solving the QAP, and numerical results indicate consistent improvement in the solution quality for all the QAP instances tested. Numerical results in Fleurent and Glover (1999) indicate that memory based modifications improve the solution quality of GRASP for solving the QAP.

Another modification of GRASP for the QAP (Li et al., 1994) can be found in Rangel et al. (1998), where a different local search scheme has been incorporated which results in a smaller number of iterations.

6.2.2. *The biquadratic assignment problem.* A generalization of the QAP is the biquadratic assignment problem (BiQAP), which is essentially a quartic assignment problem with cost coefficients formed by the products of two four-dimensional arrays. More specifically, consider two $n^4 \times n^4$ arrays $F = (f_{ijkl})$ and $D = (d_{mpst})$. The BiQAP can then be stated as:

$$\min_{p \in \Pi_N} \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} \sum_{l=1}^{n} f_{ijkl} d_{p(i)p(j)p(k)p(l)},$$

where Π_N denotes the set of all permutations of $N = \{1, 2, \ldots, n\}$ as usual. The major application of the BiQAP arises in VLSI circuit design. The majority of VLSI circuits are sequential. Their design process is composed of first translating the circuit specifications into a state transition table, modeling the system using finite state machines, and finally trying to find an encoding of the states such that the actual implementation is of minimum size. A detailed description of the mathematical modeling of the VLSI problem to a BiQAP can be found in Burkard et al. (1994). Several heuristics have been developed for the BiQAP by Burkard and Çela (1995). In particular, deterministic improvement methods and variants of simulated annealing and tabu search are proposed. Computational experiments on

test problems of size up to $n = 32$, with known optimal solutions (a test problem generator is presented in Burkard et al. (1994)), suggested that one version of simulated annealing is best among those tested. GRASP has also been applied to the BiQAP by Mavridou et al. (1998), and produced optimal solutions for all the test problems generated in Burkard et al. (1994).

6.2.3. *Multidimensional assignment problems.* In the multidimensional assignment problem (MAP), the input data consists of one cost array $C \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_M}$, where, without loss of generality, we assume that $n_1 \leq n_2 \leq \cdots \leq n_M$. An integer programming formulation of the M-dimensional assignment problem is

$$\text{(1)} \qquad \min \quad \sum_{i_1=1}^{n_1} \cdots \sum_{i_M=1}^{n_M} C_{i_1 \cdots i_M} x_{i_1 \cdots i_M}$$

$$\text{(2)} \qquad \text{s.t.} \quad \sum_{i_2=1}^{n_2} \cdots \sum_{i_M=1}^{n_N} x_{i_1 \cdots i_M} = 1, \quad i_1 = 1, \ldots, n_1,$$

$$\text{(3)} \qquad \sum_{i_1=1}^{n_1} \cdots \sum_{i_{k-1}=1}^{n_{k-1}} \sum_{i_{k+1}=1}^{n_{k+1}} \cdots \sum_{i_M=1}^{n_M} x_{i_1 \cdots i_M} = 1,$$

$$\text{(4)} \qquad \text{for } i_k = 1, \ldots, n_k \text{ and } k = 2, \ldots, M-1$$

$$\text{(5)} \qquad \sum_{i_1=1}^{n_1} \cdots \sum_{i_{M-1}=1}^{n_{M-1}} x_{i_1 \cdots i_M} = 1, \quad i_M = 1, \ldots, n_M,$$

$$\text{(6)} \qquad x_{i_1 \cdots i_M} \in \{0, 1\} \text{ for all } i_1 \cdots i_M.$$

Multidimensional assignment problems, in their general form, have found many applications as a means of modeling data association problems. More specifically, the central problem in any multitarget tracking task and multisensor surveillance is the data association problem of partitioning the observations into tracks and false alarms. General classes of these problems can be formulated as multidimensional assignment problems. For a detailed description on the application of MAPs for multiple target tracking applications see Murphey et al. (1998a). Various applications are also contributed to special cases of the MAP, where the dimension of the problem M is constant. Specifically, the five-dimensional assignment problem has been successfully used for tracking elementary particles. By solving a five-dimensional assignment problem, physicists reconstruct tracks generated by charged elementary particles produced by the large electron-positron colider (LEP) at CERN (Pusztaszeri et al., 1995). Another well known special case is the three-index assignment problem, where $M = 3$ (see Aiex et al. (2000a) for a GRASP for the three-index assignment problem).

Murphey et al. (1998a) and Murphey et al. (1998b) present a GRASP for solving the MAP for arbitrary M. In Murphey et al. (1998a), the authors focused on the mathematical modeling of the multi-target multi-sensor problem into an M-dimensional assignment problem, and present a preliminary GRASP for solving the resulting MAP, coded in MATLAB. In their following paper (Murphey et al., 1998b) they present a parallel GRASP implementation for solving the MAP, where specialized data structures used for the construction and local search phases are described in detail. In (Murphey et al., 1998b), the MAP is shown to be equivalent

to

$$\begin{aligned} \min \quad & \sum_{i=1}^{n_1} C_{ip_2(i)\cdots p_{M-1}(i)p_M(i)} \\ \text{s.t.} \quad & p_i \in \Pi_{N_i}, \quad i = 2, \ldots, M, \end{aligned}$$

where Π_{N_i} is the set of all permutations of the sets of integers $N_i = \{1, 2, \ldots, n_i\}$ for $i = 2, \ldots, M$. Experimental results in Murphey et al. (1998b) for a set of realistic test problems, indicate that GRASP produced in real-time either optimal solutions or solutions very close to the optimal partitions of observations to targets, for problems with closely spaced targets.

Robertson (2001) presents four GRASP implementations for solving the MAP, also in the context of the multi-target multi-sensor problem. Computational results are reported for two randomly generated problem sets, and a third problem set which resulted from track initiation scenarios generated from a tracking simulation engine.

6.2.4. *Plant Location.* In the capacitated plant location problem (CPLP), we have a set of alternative plants and a set of clients to be assigned to the plants. Each plant has a given capacity and a certain fixed opening cost, and each client has a given demand and a cost associated with the assignment to a plant. The objective is to find which plants to open and the assignment of clients to those opened plants at a minimum cost, such that the demands of the clients are satisfied and the capacities of the plants are not exceeded. In the case where we are constrained to assign each client to only one plant, the problem becomes the pure integer capacitated plant location problem (PI-CPLP). The PI-CPLP is also referred to as the single source CPLP in the literature. The PI-CPLP can be formulated as

$$\begin{aligned} \min \quad & \sum_{i\in I}\sum_{j\in J} c_{ij}x_{ij} + \sum_{j\in J} f_j y_j \\ \text{s.t.} \quad & \sum_{j\in J} x_{ij} = 1, \ \ \forall i \in I, \\ & \sum_{i\in I} d_i x_{ij} \le b_j y_j \ \ \forall j \in J, \\ & x_{ij}, y_j \in \{0,1\} \ \ \forall i \in I, j \in J, \end{aligned}$$

where sets I and J are the sets of clients and plants, respectively, f_i is the fixed opening cost for plant i, c_{ij} is the cost of assigning client i to plant j, d_i is the demand of client i, and b_j is the capacity of plant j. The decision variables are x_{ij} and y_j. x_{ij} is equal to 1 if client i is assigned to plant j, and 0 otherwise, and y_j is equal to 1 if plant j is open, and 0 otherwise.

Delmaire et al. (1999) develop four types of heuristics for solving the PI-CPLP, based on four different heuristic approaches: evolution algorithms (EA), GRASP, simulated annealing (SA), and tabu search (TS). In an experimental study, they compare the performance of the algorithms for solving the PI-CPLP and conclude that a hybrid approach where the speed of GRASP and the solution quality of TS are combined would be very effective. It is noted that, in (Delmaire et al., 1999), the maximum number of iterations that GRASP was allowed to run was 50, which resulted in running times which were, on average, an order of magnitude smaller

than the other heuristic procedures, while overall GRASP was second in terms of solution quality.

As a sequel to (Delmaire et al., 1999), Díaz and Fernández (1998) present a hybrid reactive GRASP and tabu search for solving the PI-CPLP. Díaz and Fernández (1998) perform computational runs on the same set of problems as in (Delmaire et al., 1999), where the hybrid GRASP-tabu search performs well. Specifically, the average deviation of the new approach, from the best known solutions never exceeded 0.3%.

Another important problem in the context of plant location, is the one which involves economies of scale. Let I and J be the sets of clients and plants respectively, as defined previously in this subsection. Each client i has a demand b_i, and the cost of shipping one unit of product from plant i to client j is c_{ij}. The location problem with economies of scale is formulated as

$$\begin{aligned}
\min \quad & \sum_{i\in I}\sum_{j\in J} c_{ij}x_{ij} + \sum_{j\in J} g_j(y_j) \\
\text{s.t.} \quad & \sum_{i\in I} x_{ij} = y_j, \ \ \forall j \in J, \\
& \sum_{j\in J} x_{ij} = b_i, \ \ \forall i \in I, \\
& y_j \geq 0, \ \ \forall j \in J, \\
& x_{ij} \geq 0, \ \ \forall i \in I, j \in J,
\end{aligned}$$

where x_{ij} is the amount of products shipped from plant j to client i, and y_j is the total amount of products shipped from plant j. The function $g(\cdot)$ is a cost function that can have several forms depending on the application. For example, in the CPLP g is constant and represents a fixed opening cost. Holmqvist et al. (1998) present a GRASP for solving the above problem where the function g has the form of warehousing costs, i.e.

$$g_j(y_j) = \begin{cases} 0 & \text{if } y_j = 0, \\ a_{1j} + a_{2j}y_j + a_{3j}\sqrt{y_j} & \text{if } y_j > 0, \end{cases}$$

where a_{1j}, a_{2j} and a_{3j} are nonnegative parameters. They exploit properties of the given problem in the GRASP implementation, and perform computational experiments for a set of problems from the OR-library (Beasley, 1990). Compared to the solutions obtained with Lagrangean relaxation, GRASP found a better solution for all but two (out of fifteen) problem instances.

6.2.5. *Frequency assignment problem.* The frequency assignment problem (FAP) arises in the context of resource management in cellular networks, and refers to the problem of assigning a limited number of frequencies to transmitters in the network, such that the electromagnetic interference is minimized. Consider that we have N transmitters in the network, and let the $N \times N$ matrix C represent the minimal frequency separation values for each pair of transmitters. Then, given two transmitters i and j and assigned frequencies f_i and f_j respectively, the following constraint should be satisfied

$$|f_i - f_j| > c_{ij}.$$

Given the minimal frequency separation matrix C in the FAP we have the following objective function

$$\min \sum_{i=1}^{N} \sum_{j=1}^{N} \delta(|f_i - f_j| > c_{ij}),$$

where

$$\delta(|f_i - f_j| > c_{ij}) = \begin{cases} c_{ij} - |f_i - f_j|, & \text{if } |f_i - f_j| \leq c_{ij}, \\ 0, & \text{otherwise.} \end{cases}$$

Liu et al. (2000) present a GRASP for solving the FAP in mobile radio networks. The construction phase is a randomized version of a sequential coloring heuristic based on saturation degrees, and the local search phase is a fast simulated annealing procedure. Computational experiments were performed on two data sets, one randomly generated and one with instances from the European EUCLID project CALMA (Combinatorial Algorithms for Military Applications). In the first data set, GRASP compares favorably with the pure saturation with the simulated annealing heuristic, while in the second set, GRASP improves the best known solutions for all instances.

6.2.6. *The p-hub location problem.* In the p-hub location problem we are given a set of nodes where there is a certain level of traffic between each pair of nodes ("traffic" can be data, products, etc.), and a cost per unit transfered between any pair of nodes. The objective is to determine p nodes which will serve as hubs, that is a set of nodes which is fully interconnected, and every other node is connected to at least one hub, such that the total cost is minimized. Given such a configuration, any two nodes in the network can communicate through the hubs (in the worst case two hubs will be used as intermediate nodes). Specifically, consider that we have n nodes, and w_{ij} is the number of units transfered from node i to node j, and c_{ij} is the cost per unit transfered between i and j. Let y_j be equal to 1 if node j is a hub and 0 otherwise, and x_{ij} be equal to 1 if node i is connected to hub j and 0 otherwise. The p-hub location problem can then be stated as

$$\begin{aligned} \min \quad & \sum_{i,j} w_{ij} \left[\sum_{k} x_{ik} c_{ik} + \sum_{m} x_{jm} c_{jm} + a \sum_{k,m} x_{ik} x_{im} c_{km} \right] \\ \text{s.t.} \quad & \sum_{j} x_{ij} = 1, \quad i = 1, \ldots, n, \\ & \sum_{j} y_j = p, \\ & x_{ij} \leq y_j, \quad i, j = 1, \ldots, n, \\ & x_{ij}, y_j \in \{0, 1\}, \quad i, j = 1, \ldots, n, \end{aligned}$$

where a is a discount parameter. Klincewicz (1992) presents a GRASP and a tabu search for solving the p-hub location problem. The author considers two classes of test problems. The first consists of instances with up to 25 nodes where $p = 3, 4$ and the optimal solutions, under the assumption that nodes are to be assigned to the nearest hub, are known. The second problem class consists of instances with 52 nodes, where $p = 4, 10$. For the first class of problems, GRASP found the optimal

solution in over 90% of the instances, while for the second problem class it generated best-known solutions.

7. Concluding remarks

GRASP has found its way into many applications in academic as well as industrial settings. Besides producing good quality solutions to hard combinatorial optimization problems, it has several advantages over other metaheuristics. It can usually be implemented quickly, since construction and local search algorithms are readily available. Parameter tuning is minimal. Finally, because of the probability distribution of its running time to find a suboptimal target solution, GRASP can be implemented in parallel with optimal speed-up.

Many problems can be approached with GRASP. However, since GRASP is a constructive method, it is difficult to apply it to problems for which feasible solutions are difficult to construct.

References

A.V. Aho, J.E. Hopcroft, and J.D. Ullman. *The design and analysis of computer algorithms*. Addison-Wesley, 1974.

R. K. Ahuja, J. B. Orlin, and D. Sharma. New neighborhood search structures for the capacitated minimum spanning tree problem. Technical report, Department of ISE, University of Florida, Gainesville, FL 32611-6595, 1998.

R.K. Ahuja, J.B. Orlin, and A. Tiwari. A greedy genetic algorithm for the quadratic assignment problem. *Computers and Operations Research*, 27:917–934, 2000.

R.M. Aiex, M.G.C. Resende, P.M. Pardalos, and G. Toraldo. GRASP with path relinking for the three-index assignment problem. Technical report, AT&T Labs Research, Florham Park, NJ 07932 USA, 2000a.

R.M. Aiex, M.G.C. Resende, and C.C. Ribeiro. Probability distribution of solution time in GRASP: An experimental investigation. Technical report, AT&T Labs Research, Florham Park, NJ 07733, 2000b. Forthcoming in *Journal of Heuristics*.

J. E. Beasley. OR-Library: Distributing test problems by electronic mail. *Journal of the Operational Research Society*, 41:1069–1072, 1990.

S. Binato, W.J. Hery, D. Loewenstern, and M.G.C. Resende. A greedy randomized adaptive search procedure for job shop scheduling. In P. Hansen and C.C. Ribeiro, editors, *Essays and surveys on metaheuristics*. Kluwer Academic Publishers, 2001.

J.L. Bresina. Heuristic-biased stochastic sampling. In *Proceedings of the AAAI-96*, pages 271–278, 1996.

R. E. Burkard, S. Karisch, and F. Rendl. QAPLIB – A quadratic assignment problem library. *Journal of Global Optimization*, 10:391–403, 1997. An on-line version is available via World Wide Web at the following URL: `http://www.opt.math.tu-graz.ac.at/qaplib/`.

R.E. Burkard and E. Çela. Heuristics for biquadratic assignment problems and their computational comparison. *European Journal of Operational Research*, 83: 283–300, 1995.

R.E. Burkard, E. Çela, and B. Klinz. On the biquadratic assignment problem. In P.M. Pardalos and H. Wolkowicz, editors, *Quadratic assignment and related problems*, DIMACS Series on Discrete Mathematics and Theoretical Computer Science, pages 117–146. American Mathematical Society, 1994.

R. Colomé and D. Serra. Consumer choice in competitive location models: Formulations and heuristics. Technical report, Department of Economics and Management, Universitat Pompeu Fabra, Barcelona, Spain, 1998. Forthcoming in *Papers in Regional Science*.

S.A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the Third annual ACM Symposium on Theory of Computing*, pages 151–158, 1971.

T.H. Cormen, C.E. Leiserson, and R.L. Rivest. *Introduction to algorithms*. The MIT Press, 1990.

H. Delmaire, J.A. Díaz, E. Fernández, and M. Ortega. Reactive GRASP and Tabu Search based heuristics for the single source capacitated plant location problem. *INFOR*, 37:194–225, 1999.

A.S. Deshpande and E. Triantaphyllou. A greedy randomized adaptive search procedure (GRASP) for inferring logical clauses from examples in polynomial time and some extensions. *Mathl. Comput. Modelling*, 27:75–99, 1998.

J.A. Díaz and E. Fernández. A hybrid GRASP-Tabu Search algorithm for the single source capacitated plant location problem. Technical report, Department of Statistics and Operations Research, Universitat Politecnica de Catalunya, Barcelona, Spain, 1998.

A. N. Elshafei. Hospital layout as a quadratic assignment problem. *Operations Research Quarterly*, 28:167–179, 1977.

T.A. Feo and J.L. González-Velarde. The intermodal trailer assignment problem: Models, Algorithms, and Heuristics. *Transportation Science*, 29:330–341, 1995.

T.A. Feo and M.G.C. Resende. A probabilistic heuristic for a computationally difficult set covering problem. *Operations Research Letters*, 8:67–71, 1989.

T.A. Feo and M.G.C. Resende. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6:109–133, 1995.

E. Fernández and R. Martí. GRASP for seam drawing in mosaicking of aerial photographic maps. *Journal of Heuristics*, 5:181–197, 1999.

P. Festa and M.G.C. Resende. GRASP: An annotated bibliography. In P. Hansen and C.C. Ribeiro, editors, *Essays and Surveys on Metaheuristics*. Kluwer Academic Publishers, 2001.

C. Fleurent and F. Glover. Improved constructive multistart strategies for the quadratic assignment problem using adaptive memory. *INFORMS Journal on Computing*, 11:198–204, 1999.

M.R. Garey and D.S. Johnson. *Computers and intractability - A guide to the theory of NP-completeness*. W.H. Freeman and Company, 1979a.

M.R. Garey and D.S. Johnson. *Computers and intractability: A guide to the theory of NP-completeness*. W.H. Freeman and Company, New York, 1979b.

A. M. Geoffrion and G. W. Graves. Scheduling parallel production lines with changeover costs: Practical applications of a quadratic assignment/LP approach. *Operations Research*, 24:595–610, 1976.

F. Glover and M. Laguna. *Tabu Search*. Kluwer Academic Publishers, 1997.

F. Glover, M. Laguna, and R. Martí. Fundamentals of scatter search and path relinking. Technical report, Graduate School of Business and Administration, University of Colorado, Boulder, CO 80309-0419, 2000.

P. Hansen and N. Mladenović. An introduction to variable neighborhood search. In S. Voss, S. Martello, I. H. Osman, and C. Roucairol, editors, *Meta-heuristics, Advances and trends in local search paradigms for optimization*, pages 433–458. Kluwer Academic Publishers, 1998.

J.P. Hart and A.W. Shogan. Semi-greedy heuristics: An empirical study. *Operations Research Letters*, 6:107–114, 1987.

D. R. Heffley. Assigning runners to a relay team. In S. P. Ladany and R. E. Machol, editors, *Optimal Strategies in Sports*, pages 169–171. North-Holland, Amsterdam, 1977.

K. Holmqvist, A. Migdalas, and P.M. Pardalos. A GRASP algorithm for the single source uncapacitated minimum concave-cost network flow problem. In P.M. Pardalos and D.-Z. Du, editors, *Network design: Connectivity and facilities location*, volume 40 of *DIMACS Series on Discrete Mathematics and Theoretical Computer Science*, pages 131–142. American Mathematical Society, 1998.

D.S. Johnson, C.H. Papadimitriou, and M. Yannakakis. How easy is local search? *Journal of Computer and System Sciences*, 17:79–100, 1988.

D.S. Johnson and M.A. Trick, editors. *Cliques, coloring, and Satisfiability: Second DIMACS Implementation Challenge*. DIMACS Series in Discrete Mathematics and Theoretical Computer Science. American Mathematical Society, 1996.

J.G. Klincewicz. Avoiding local optima in the p-hub location problem using tabu search and GRASP. *Annals of Operations Research*, 40:283–302, 1992.

T. C. Koopmans and M. J. Beckmann. Assignment problems and the location of economic activities. *Econometrica*, 25:53–76, 1957.

J. Krarup and P. M. Pruzan. Computer-aided layout design. *Mathematical Programming Study*, 9:75–94, 1978.

M. Laguna and J.L. González-Velarde. A search heuristic for just-in-time scheduling in parallel machines. *Journal of Intelligent Manufacturing*, 2:253–260, 1991.

M. Laguna and R. Martí. GRASP and path relinking for 2-layer straight line crossing minimization. *INFORMS J. on Computing*, 11:44–52, 1999.

Y. Li, P.M. Pardalos, and M.G.C. Resende. A greedy randomized adaptive search procedure for the quadratic assignment problem. In P.M. Pardalos and H. Wolkowicz, editors, *Quadratic assignment and related problems*, volume 16 of *DIMACS Series on Discrete Mathematics and Theoretical Computer Science*, pages 237–261. American Mathematical Society, 1994.

X. Liu, P.M. Pardalos, S. Rajasekaran, and M.G.C. Resende. A GRASP for frequency assignment in mobile radio networks. In S. Rajasekaran, P.M. Pardalos, and F. Hsu, editors, *Mobile Networks and Computing*, volume 52 of *DIMACS Series on Discrete Mathematics and Theoretical Computer Science*, pages 195–201. American Mathematical Society, 2000.

H. Ramalhinho Lourenço, J.P. Paixão, and R. Portugal. Metaheuristics for the bus-driver scheduling problem. Technical report, Department of Economics and Management, Universitat Pompeu Fabra, Barcelona, Spain, 1998.

H. Ramalhinho Lourenço and D. Serra. Adaptive approach heuristics for the generalized assignment problem. Technical report, Department of Economics and Management, Universitat Pompeu Fabra, Barcelona, Spain, 1998.

S.L. Martins, M.G.C. Resende, C.C. Ribeiro, and P.M. Pardalos. A parallel GRASP for the Steiner tree problem in graphs using a hybrid local search strategy. *Journal of Global Optimization*, pages 267–283, 2000.

S.L. Martins, C.C. Ribeiro, and M.C. Souza. A parallel GRASP for the Steiner problem in graphs. In A. Ferreira and J. Rolim, editors, *Proceedings of IRREGUALAR'98 – 5th International Symposium on Solving Irregularly Structured Problems in Parallel*, volume 1457 of *Lecture Notes in Computer Science*, pages 285–297. Springer-Verlag, 1998.

T. Mavridou, P.M. Pardalos, L.S. Pitsoulis, and M.G.C. Resende. A GRASP for the biquadratic assignment problem. *European J. of Operational Research*, 105: 613–621, 1998.

J. Mockus, E. Eddy, A. Mockus, L. Mockus, and G.V. Reklaitis. *Bayesian discrete and global optimization*. Kluwer Academic Publishers, 1997.

R.A. Murphey, P.M. Pardalos, and L.S. Pitsoulis. A greedy randomized adaptive search procedure for the multitarget multisensor tracking problem. In P.M. Pardalos and D.-Z. Du, editors, *Network design: Connectivity and facilities location*, volume 40 of *DIMACS Series on Discrete Mathematics and Theoretical Computer Science*, pages 277–301. American Mathematical Society, 1998a.

R.A. Murphey, P.M. Pardalos, and L.S. Pitsoulis. A parallel GRASP for the data association multidimensional assignment problem. In P.M. Pardalos, editor, *Parallel processing of discrete problems*, volume 106 of *The IMA Volumes in Mathematics and Its Applications*, pages 159–180. Springer-Verlag, 1998b.

P.M. Pardalos, L.S. Pitsoulis, and M.G.C. Resende. A parallel GRASP implementation for the quadratic assignment problem. In A. Ferreira and J. Rolim, editors, *Parallel Algorithms for Irregularly Structured Problems – Irregular'94*, pages 111–130. Kluwer Academic Publishers, 1995.

P.M. Pardalos, L.S. Pitsoulis, and M.G.C. Resende. A parallel GRASP for MAX-SAT problems. *Lecture Notes in Computer Science*, 1184:575–585, 1996.

P.M. Pardalos, L.S. Pitsoulis, and M.G.C. Resende. Algorithm 769: Fortran subroutines for approximate solution of sparse quadratic assignment problems using GRASP. *ACM Transactions on Mathematical Software*, 23:196–208, 1997.

M. Prais and C.C. Ribeiro. Reactive GRASP: An application to a matrix decomposition problem in TDMA traffic assignment. *INFORMS Journal on Computing*, 12:164–176, 2000.

J. Pusztaszeri, P.E. Rensing, and T.M. Liebling. Tracking elementary particles near their primary vertex: A combinatorial approach. *Journal of Global Optimization*, 16:422–431, 1995.

M.C. Rangel, N.M.M. de Abreu, P.O. Boaventura Netto, and M.C.S. Boeres. A modified local search for GRASP in the quadratic assignment problem. Technical report, Production Engineering Program, COPPE, Federal University of Rio de Janeiro, Rio de Janeiro, RJ Brazil, 1998.

M.G.C. Resende and T.A. Feo. A GRASP for satisfiability. In D.S. Johnson and M.A. Trick, editors, *Cliques, Coloring, and Satisfiability: The Second DIMACS Implementation Challenge*, volume 26 of *DIMACS Series on Discrete Mathematics and Theoretical Computer Science*, pages 499–520. American Mathematical Society, 1996.

M.G.C. Resende, P.M. Pardalos, and Y. Li. Algorithm 754: Fortran subroutines for approximate solution of dense quadratic assignment problems using GRASP. *ACM Transactions on Mathematical Software*, 22:104–118, 1996.

M.G.C. Resende, L.S. Pitsoulis, and P.M. Pardalos. Approximate solution of weighted MAX-SAT problems using GRASP. In J. Gu and P.M. Pardalos, editors, *Satisfiability problems*, volume 35 of *DIMACS Series on Discrete Mathematics and Theoretical Computer Science*, pages 393–405. American Mathematical Society, 1997.

M.G.C. Resende, L.S. Pitsoulis, and P.M. Pardalos. Fortran subroutines for computing approximate solutions of MAX-SAT problems using GRASP. *Discrete Applied Mathematics*, 100:95–113, 2000.

M.G.C. Resende and O. Ulular. SMART: A tool for AT&T Worldnet access design – Location of Cascade 9000 concentrators. Technical report, AT&T Labs Research, Florham Park, NJ 07932 USA, 1997.

A.J. Robertson. A set of greedy randomized adaptive local search procedure (GRASP) implementations for the multidimensional assignment problem. *Computational Optimization and Applications*, 2001.

S. Sahni and T. Gonzalez. P-complete approximation problems. *Journal of the Association of Computing Machinery*, 23:555–565, 1976.

B. Selman, H. Levesque, and D. Mitchell. A new method for solving hard satisfiability problems. In *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI92)*, pages 440–446, 1992.

I. Ugi, J. Bauer, J. Friedrich, J. Gasteiger, C. Jochum, and W. Schubert. Neue anwendungsgebiete fur computer in der chemie. *Angew. Chemie*, 91:99–184, 1979.

M.G.A. Verhoeven and E.H.L. Aarts. Parallel local search. *J. of Heuristics*, 1: 43–66, 1995.

(Leonidas S. Pitsoulis) Department of Operations Research and Financial Engineering, Princeton University, Princeton, NJ 08544 USA.

E-mail address: leonidas@dragon.princeton.edu

(Mauricio G. C. Resende) Information Sciences Research Center, AT&T Labs Research, 180 Park Avenue, Room C241, Florham Park, NJ 07932 USA.

E-mail address: mgcr@research.att.com