GRASP: Greedy randomized adaptive search procedures

Mauricio G. C. Resende
Algorithms and Optimization Research Department
AT&T Labs – Research
180 Park Avenue
Florham Park, NJ 07732 USA
mgcr@research.att.com

Jośe Luis González Velarde
Tecnológico de Monterrey
Garza Sada 2501
Monterrey, NL 64849 Mexico
lugonzal@campus.mty.itesm.mx

27 de febrero de 2003

1. Introduction

Consider a combinatorial optimization problem defined by a finite ground set $E = \{1, \dots, n\}$, a set of feasible solutions $F \subseteq 2^E$, and an objective function $f : 2^E \to \mathbb{R}$. In the minimization version, we seek an optimal solution $S^* \in F$ such that $f(S^*) \leq f(S)$, $\forall S \in F$. The ground set E, the cost function f, as well as the set of feasible solutions F are defined for each specific problem. For instance, in the case of the traveling salesman problem, the ground set E is that of all edges connecting the cities to be visited, $f(S)$ is the sum of the costs of all edges $e \in S$, and F is formed by all edge subsets that determine a Hamiltonian cycle.

A greedy randomized adaptive search procedure (GRASP) [46, 47] is a metaheuristic for finding approximate (i.e. good sub-optimal, but not necessarily optimal) solutions to combinatorial optimization problems. It is based on the premise that diverse, good-quality starting solutions play an important role in the success of local search methods.

A GRASP is a multi-start method, in which each GRASP iteration consists of the construction of a randomized greedy solution followed by local search using the constructed solution as the starting point for local search. This procedure is applied repeated times and the best solution found over all GRASP iterations is returned as the approximate solution. The pseudo-code in Figure 1 illustrates a basic GRASP for minimization.

procedure `GRASP`
Require: $i_{m\acute{a}x}$
  $f^* \leftarrow \infty$;
  for $i \leq i_{m\acute{a}x}$ **do**
    $x \leftarrow$ `GreedyRandomized`();
    $x \leftarrow$ `LocalSearch`(x);
    if $f(x) < f^*$ **then**
      $f^* \leftarrow f(x)$;
      $x^* \leftarrow x$;
    end if
  end for
  return x^*;

Figura 1: GRASP pseudo-code

In this paper, we first focus on the two most important components of GRASP, namely construction and local search. Then we examine how path-relinking can be used in GRASP as a memory and intensification mechanism. Parallel GRASP is discussed next. The paper ends with a partial list of successful GRASP applications.

Recent surveys on GRASP can be found in [99, 90] and an extensive annotated bibliography is in [54] and is updated at the URL `http://graspheuristic.org/annotated`.

```
procedure Construct-C
Require: k, E, c(·);
  x ← ∅;
  C ← E;
  Compute greedy cost c(e), ∀ e ∈ C;
  while C ≠ ∅ do
    RCL ← {k elements e ∈ C with smallest
    c(e)};
    Select element s at random from RCL;
    x ← x ∪ {s};
    Update candidate set C;
    Compute greedy cost c(e), ∀ e ∈ C;
  end while
  return x;
```

Figura 2: GRASP construction pseudo-code: cardinality based RCL

```
procedure Construct-V
Require: α, E, c(·);
  x ← ∅;
  C ← E;
  Compute greedy cost c(e), ∀ e ∈ C;
  while C ≠ ∅ do
    c_* ← mín{c(e) | e ∈ C};
    c^* ← máx{c(e) | e ∈ C};
    RCL ← {e ∈ C | c(e) ≤ c_* + α(c^* − c_*)};
    Select element s at random from RCL;
    x ← x ∪ {s};
    Update candidate set C;
    Compute greedy cost c(e), ∀ e ∈ C;
  end while
  return x;
```

Figura 3: GRASP construction pseudo-code: value-base RCL

2. GRASP construction

In this section we describe several greedy randomized construction mechanisms. These procedures mix greediness with randomization in different ways.

All of the construction mechanisms that we consider build a solution one element at a time. At each step of the construction process, a partial solution is on hand. An element that can be selected to be part of a partial solution is called a *candidate* element. Consider a set covering problem, where one is given a matrix $A = [a_{ij}]$ of zeros and ones, a cost c_j for each column j, and wants to determine a set J of columns having the smallest total cost $\sum_{j \in J} c_j$ such that for each row i, at least one column j of the set (cover) has entry $a_{ij} = 1$. In this problem, a partial solution is a set of columns not necessarily forming a cover. Any previously unselected column is a candidate element. The solution set J is built one element (column) at a time until set J is a cover.

To determine which candidate element to select next to be included in the solution, one usually makes use of a greedy function. A greedy function measures the myopic contribution of each element to the partial solution. In the case of set covering, a sensible greedy function is the ratio p_j/c_j between the number p_j of yet-uncovered rows that would become covered if column j is selected and the contribution c_j to the total cost of selecting column j to be in the solution. The greedy choice would be to add the column with the largest greedy function.

There are several possible ways to add randomness to this procedure. One of the first ideas was the use of a restricted candidate list (RCL) [46]. Such a list contains a set of candidate elements with high greedy function values. The next candidate to be added to the solution is selected at random from the restricted candidate list. The RCL can consist of a fixed number of elements (cardinality restriction) or elements with greedy function values within a given range. Figure 2 shows pseudo-code for a GRASP construction procedure based on cardinality restriction. For example, let c^* and c_* denote, respectively, the largest and smallest greedy function values for the candidate elements, and let α be a real number such that $0 \leq \alpha \leq 1$. In a value-based restricted candidate list, the RCL consists of all candidate elements e whose greedy function value $c(e)$ is such that $c(e) \leq c_* + \alpha(c^* - c_*)$. Note that if $\alpha = 0$, then this selection scheme is a greedy algorithm, whereas if $\alpha = 1$, then it is totally random. Figure 3 shows pseudo-code for a GRASP construction procedure based on value restriction. Determining which value of α to use will be discussed later.

One can also mix random construction with greedy construction as follows. Select a partial set of candidate elements sequentially at random and then complete the solution using a greedy algorithm [101]. Figure 4 shows pseudo-code for such a construction procedure.

Another approach is through cost perturbation. In this, one randomly perturbs the cost data and applies a greedy algorithm [30]. Figure 5 shows

procedure `Construct-RG`
Require: k, E, $c(\cdot)$;
$x \leftarrow \emptyset$;
$C \leftarrow E$;
Compute greedy cost $c(e)$, $\forall\, e \in C$;
for $i = 1, 2, \ldots, k$ **do**
  if $C \neq \emptyset$ **then**
    Select element e at random from C;
    $x \leftarrow x \cup \{e\}$;
    Update candidate set C;
    Compute greedy cost $c(e)$, $\forall\, e \in C$;
  end if
end for
while $C \neq \emptyset$ **do**
  $e_* \leftarrow \text{argmin}\{c(e) \mid e \in C\}$;
  $x \leftarrow x \cup \{e_*\}$;
  Update candidate set C;
  Compute greedy cost $c(e)$, $\forall\, e \in C$;
end while
return x;

Figura 4: GRASP construction pseudo-code: random then greedy construction

procedure `Construct-PG`
Require: E, $c(\cdot)$;
$x \leftarrow \emptyset$;
$C \leftarrow E$;
Randomly perturb problem data;
Compute perturbed greedy cost $\tilde{c}(e)$, $\forall\, e \in C$;
while $C \neq \emptyset$ **do**
  $e_* \leftarrow \text{argmin}\{\tilde{c}(e) \mid e \in C\}$;
  $x \leftarrow x \cup \{e_*\}$;
  Update candidate set C;
  Compute perturbed greedy cost $\tilde{c}(e)$, $\forall\, e \in C$;
end while
return x;

Figura 5: GRASP construction pseudo-code: construction with perturbations

procedure `Construct-B`
Require: α, E, $c(\cdot)$;
$x \leftarrow \emptyset$;
$C \leftarrow E$;
Compute greedy cost $c(e)$, $\forall\, e \in C$;
while $C \neq \emptyset$ **do**
  $c_* \leftarrow \text{mín}\{c(e) \mid e \in C\}$;
  $c^* \leftarrow \text{máx}\{c(e) \mid e \in C\}$;
  $\text{RCL} \leftarrow \{e \in C \mid c(e) \leq c_* + \alpha(c^* - c_*)\}$;
  Assign rank $r(e)$, $\forall\, e \in RCL$;
  Assign a probability $\pi(r(e))$ of selecting element e RCL favoring well ranked candidate elements;
  Select element s at random from RCL with probability $\pi(r(s))$;
  $x \leftarrow x \cup \{s\}$;
  Update candidate set C;
  Compute greedy cost $c(e)$, $\forall\, e \in C$;
end while
return x;

Figura 6: GRASP construction pseudo-code: biased value-based RCL

pseudo-code for this construction procedure.

A final example of a GRASP construction procedure is a variation on the value-based RCL approach. In this procedure, called bias function [28], instead of selecting the element from the RCL at random with equal probability assigned to each element, different probabilities are assigned, favoring well-ranked elements, i.e. elements with low greedy cost over elements with higher costs. The elements of the RCL are ranked according to their greedy function values. The probability $\pi(r(\sigma))$ of selecting element σ is

$$\pi(r(\sigma)) = \frac{\texttt{bias}(r(\sigma))}{\sum_{\sigma' \in \text{RCL}} \texttt{bias}(r(\sigma'))},$$

where $r(\sigma)$ is the rank of element σ in the RCL. Several alternative for assigning bias to the elements have been proposed. For example,

- random bias: $\texttt{bias}(r) = 1$;
- linear bias: $\texttt{bias}(r) = 1/r$;
- exponential bias: $\texttt{bias}(r) = e^{-r}$.

The pseudo-code in Figure 6 illustrates this procedure.

```
procedure LocalSearch
Require: x^0, N(·), f(·);
  x ← x^0;
  while x is not locally optimal w.r.t. N(x) do
    Let y ∈ N(x) be such that f(y) < f(x);
    x ← y;
  end while
  return x;
```

Figura 7: Local search pseudo-code

In the next section, we discuss how to determine which value of α to use in the RCL-based schemes discussed above. Recall that that if $\alpha = 0$, then these selection schemes resemble a greedy algorithm, whereas if $\alpha = 1$, they are totally random.

3. Local search

A local search algorithm repeatedly explores a solution neighborhood in search for a better solution. When no improving solution is found, the solution is said to be locally optimal. Figure 7 shows pseudo-code for a local search procedure.

Local search plays an important role in GRASP seeking locally optimal solutions in promising regions of the solution space. It differentiates GRASP from the semi-greedy algorithm of Hart and Shogan [59], by definition never doing worse than semi-greedy, and almost always producing better solutions in less time.

Though greedy algorithms can produce reasonable good solutions, and starting with such a solution will usually cause local search to quickly converge to a local minimum, their main drawback as a generator of starting solutions for local search is that they lack diversity. By repeatedly applying a greedy algorithm only a single or very few solutions are generated. On the other hand, a totally random algorithm produces a large amount of diverse solutions. However, these solutions are usually of very poor quality and using them as initial solutions for local search usually leads to slow convergence to a local minimum.

To benefit from the fast convergence of the greedy algorithm and the large diversity of the random algorithm, one customarily uses an α value strictly in the interior of the range $[0,1]$. Since one does not know a priori which value to use, a reasonable strategy is to select a different value at each GRASP iteration at random. This can be done using a uniform probability [97] or using the *reactive* GRASP scheme [91].

In the reactive GRASP scheme, let $\Psi = \{\alpha_1, \ldots, \alpha_m\}$ be the set of possible values for α. The probabilities associated with the choice of each value are all initially made equal to $p_i = 1/m$, $i = 1, \ldots, m$. Furthermore, let z^* be the incumbent solution and let A_i be the average value of all solutions found using $\alpha = \alpha_i$, $i = 1, \ldots, m$. The selection probabilities are periodically reevaluated by taking $p_i = q_i / \sum_{j=1}^{m} q_j$, with $q_i = z^*/A_i$ for $i = 1, \ldots, m$. The value of q_i will be larger for values of $\alpha = \alpha_i$ leading to the best solutions on average. Larger values of q_i correspond to more suitable values for the parameter α. The probabilities associated with these more appropriate values will then increase when they are reevaluated.

More elaborate local search schemes have been used in the GRASP framework, including tabu search [67, 37, 1, 108], simulated annealing [71], variable neighborhood search [31, 53], and extended neighborhood search [3].

4. Path-relinking

Perhaps one of the main drawbacks of the pure GRASP is its lack of memory structures. GRASP iterations are independent and make no use of observations made during earlier iterations. One remedy for this is the use of path-relinking with GRASP. Path-relinking was originally proposed by Glover [57] as a way to explore trajectories between elite solutions obtained by tabu search or scatter search. Using one or more elite solutions, *paths* in the solution space leading to other elite solutions are explored to search for better solutions. To generate paths, moves are selected to introduce attributes in the current solution that are present in the elite guiding solution.

Path-relinking in the context of GRASP was introduced by Laguna and Mart´ı [68] and was followed by a number of extensions, improvements, and successful applications [5, 30, 100, 103, 101, 53]. It has been used as an intensification scheme, where solutions generated at each GRASP iteration are relinked with one or more solutions from an elite set

```
procedure PR
Require: x_s, x_t;
  Compute symmetric difference Δ(x_s, x_t);
  x ← x_s;
  f* ← mín{f(x_s), f(x_t)};
  x* ← argmin{f(x_s), f(x_t)};
  while Δ(x_s, x_t) ≠ ∅ do
    m* ← argmin{f(x ⊕ m), ∀m ∈ Δ(x, x_t)};
    Δ(x ⊕ m*, x_t) ← Δ(x, x_t) \ {m*};
    x ← x ⊕ m*;
    if f(x) < f* then
      f* ← f(x);
      x* ← x;
    end if
  end while
  return x*;
```

Figura 8: Path-relinking pseudo-code

of solutions, or in a post-optimization phase, where pairs of elite set solutions are relinked.

Consider two solutions x_s and x_t on which we wish to apply path-relinking from x_s to x_t. Figure 8 illustrates the path-relinking procedure with pseudo-code. The procedure starts by computing the symmetric difference $\Delta(x_s, x_t)$ between the two solutions, i.e. the set of moves needed to reach x_t from x_s. A *path* of solutions is generated linking x_s and x_t. The best solution in this path is returned by the algorithm. At each step, the procedure examines all moves $m \in \Delta(x, x_t)$ from the current solution x and selects the one which results in the least cost solution, i.e. the one which minimizes $f(x \oplus m)$, where $x \oplus m$ is the solution resulting from applying move m to solution x. The best move m^* is made producing $x \oplus m^*$ and move m^* is removed from the symmetric difference of $\Delta(x \oplus m^*, x_t)$. If necessary, the best solution x^* is updated. The procedure terminates when x_t is reached, i.e. when $\Delta(x, x_t) = \emptyset$.

Path-relinking maintains an elite set P of solutions found during the optimization [55]. The first $|P|$ distinct solutions found are inserted into the elite set. After that, a candidate solution x^* is added to P if its cost is smaller than the cost of all elite set solutions, or if its cost is greater than the best, but smaller than the worst elite solution and it is sufficiently different from all elite set solutions. If accepted into the elite set, the new solution replaces the solution most similar to it from the set of elite solutions having worse cost than it [101]. The elite set can be periodically renewed [4] if no change in the elite set is observed for a specified number of GRASP iterations. One way to do this is to set the objective function values of the worse half of the elite set to infinity. This way new elite set solutions will be created.

Several alternative schemes have been proposed for path-relinking. Since path-relinking can be computationally demanding, it need not be applied after every GRASP iteration, but rather periodically. Usually the paths from x_s to x_t and from x_t to x_s are different and both can be explored. Since paths can be long, the full trajectory need not be followed. One can restrict following a truncated path starting at x_s and another starting at x_t.

5. Parallel GRASP

Most parallel implementations of GRASP follow the *multiple-walk independent thread* strategy [112], which distributes the iterations over the processors [4, 8, 9, 48, 70, 76, 78, 82, 87, 88]. In general, each search thread has to perform $i_{máx}/p$ iterations, where p is the number of processors and $i_{máx}$ is total number of iterations. Each processor keeps a copy of the sequential algorithm, the problem data, and an distinct seed to generate a pseudo-random number stream. A single global variable is used to store the best solution found over all processors. One of the processors is the master, reading and distributing problem data, generating the seeds used by the pseudo-random number generators at each processor, distributing the iterations, and collecting the best solution found by each processor. Since the iterations are independent and a small amount of information is exchanged, linear speedups can be easily obtained if no major load imbalance is present.

Martins et al. [78] implemented a parallel GRASP for the Steiner problem in graphs. Parallelization is achieved by the distribution of 512 iterations over the processors, with the value of the RCL parameter α randomly chosen in the interval $[0{,}0, 0{,}3]$ at each iteration. The algorithm was implemented in C on an IBM SP-2 machine with 32 processors, using the MPI library for communication. The 60 problems from series C, D, and E of the OR-Library [23] were used in the computational experiments. The parallel implementation obtained 45 optimal solutions over the 60 test instances. The relative deviation with respect to the optimal value was never larger than 4%. Almost-linear speedups observed for 2, 4, 8, and 16 processors with respect to the sequential implementation are illustrated in Figure 9.

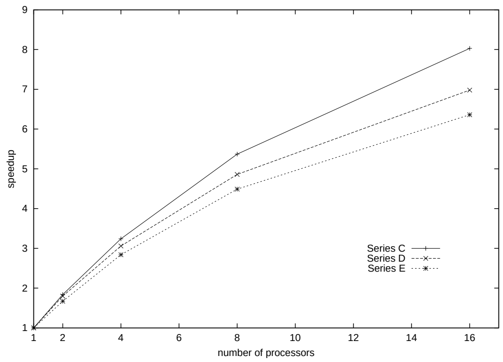

Figura 9: Average speedups on 2, 4, 8, and 16 processors.

Path-relinking may also be used in conjunction with parallel implementations of GRASP. In the case of the multiple-walk independent-thread implementation described by Aiex et al. [5] for the 3-index assignment problem, each processor applies path-relinking to pairs of elite solutions stored in a local pool. Computational results using MPI on an SGI Challenge computer with 28 R10000 processors showed linear speedups.

Alvim and Ribeiro [8, 9] showed that multiple-walk independent-thread approaches for the parallelization of GRASP may benefit from load balancing techniques whenever heterogeneous processors are used or if the parallel machine is simultaneously shared by several users. Almost-linear speedups can be obtained with a heterogeneous distribution of the iterations over the p processors in $q \geq p$ packets. Each processor starts performing a packet of $\lceil i_{\text{máx}}/q \rceil$ iterations and informs the master when it is done with its packet of iterations. The master stops the execution of each slave processor when there are no more iterations to be performed and collects the best solution found. Faster or less loaded processors will perform more iterations than the others. In the case of the parallel GRASP implemented for the traffic assignment problem described in [91], this type of dynamic load balancing strategy allowed reductions in the elapsed times of up to 15 % with respect to the times observed for the static strategy, in which the iterations were uniformly distributed over the processors.

The efficiency of multiple-walk independent-thread parallel implementations of metaheuristics, based on running multiple copies of the same sequential algorithm, has been addressed by several authors. A given target value τ for the objective function is broadcast to all processors, which independently execute the sequential algorithm. All processors halt immediately after one of them finds a solution with value at least as good as τ. The speedup is given by the ratio between the times needed to find a solution with value at least as good as τ, using respectively the sequential algorithm and the parallel implementation with p processors. It is necessary to ensure that no two iterations start with identical random number generator seeds. These speedups are linear for a number of metaheuristics, including simulated annealing [39, 84], iterated local search algorithms for the traveling salesman problem [41], tabu search, provided that the search starts from a local optimum [22, 109], and WalkSAT [107] on hard random 3-SAT problems [61]. This observation can be explained if the random variable *time to find a solution at least as good some target value* is exponentially distributed, as indicated by the following proposition [112]:

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

This proposition follows from the definition of the exponential distribution. It implies that the probability $1 - e^{-\rho t/\lambda}$ of finding a solution within a given target value in time ρt with a sequential algorithm is equal to the probability of finding a solution at least as good as that in time t using ρ independent parallel processors. Hence, it is possible, on average, to achieve linear speedups in the time to find a solution within a target value by multiple independent processors. An analogous proposition can be stated for a two parameter (shifted) exponential distribution [6]:

Proposition 2: Let $P_\rho(t)$ be the probability of not having found a given target solution value in t time units with ρ independent processors. If $P_1(t) = e^{-(t-\mu)/\lambda}$ with $\lambda \in \mathbb{R}^+$ and $\mu \in \mathbb{R}^+$, corresponding 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 within a given target value in time ρt with a sequential algorithm is equal to $1 - e^{-(\rho t - \mu)/\lambda}$, while the probability of finding a solution at least as good as that in time t using ρ independent parallel pro-

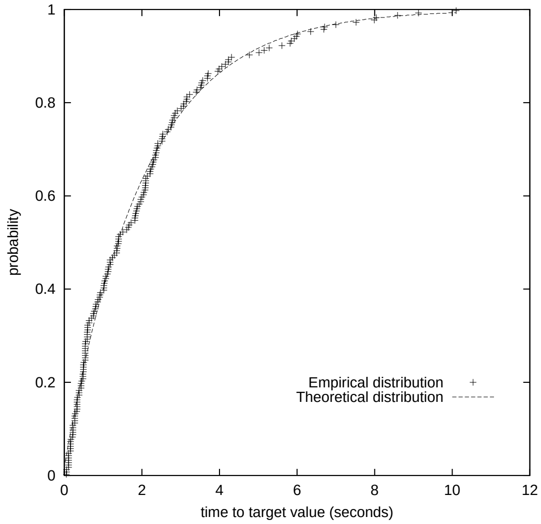

Figura 10: Superimposed empirical and theoretical distributions (times to target values measured in seconds on an SGI Challenge computer with 28 processors).

Cuadro 1: Speedup with respect to a single processor implementation. Algorithms are independent and cooperative implementations of GRASP with path-relinking. Instances are `abz6`, `mt10`, `orb5`, and `la21`, with target values 943, 938, 895, and 1100, respectively.

independent parallel GRASP

	processors			
problem	2	4	8	16
`abz6`	2.00	3.36	6.44	10.51
`mt10`	1.57	2.12	3.03	4.05
`orb5`	1.95	2.97	3.99	5.36
`la21`	1.64	2.25	3.14	3.72
average:	**1.79**	**2.67**	**4.15**	**5.91**

cooperative parallel GRASP

	processors			
problem	2	4	8	16
`abz6`	2.40	4.21	11.43	23.58
`mt10`	1.75	4.58	8.36	16.97
`orb5`	2.10	4.91	8.89	15.76
`la21`	2.23	4.47	7.54	11.41
average:	**2.12**	**4.54**	**9.05**	**16.93**

cessors is $1 - e^{-\rho(t-\mu)/\lambda}$. 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 speedups in the time to find a solution within a target value using multiple independent processors.

Aiex et al. [6] have shown experimentally that the solution times for GRASP also have this property, showing that they fit a two-parameter exponential distribution. Figure 10 illustrates this result, depicting the superimposed empirical and theoretical distributions observed for one of the cases studied along the computational experiments reported by the authors, which involved 2400 runs of GRASP procedures for each of five different problems: maximum independent set [48, 94], quadratic assignment [70, 95], graph planarization [98, 102], maximum weighted satisfiability [97], and maximum covering [92]. The same result still holds when GRASP is implemented in conjunction with a post-optimization path-relinking procedure [5, 4].

In the case of *multiple-walk cooperative-thread* strategies, the search threads running in parallel exchange and share information collected along the trajectories they investigate. One expects not only to speed up the convergence to the best solution but, also, to find better solutions than those found by independent-thread strategies. The most difficult aspect to be set up is the determination of the nature of the information to be shared or exchanged to improve the search. Care is needed not to use up too much additional memory or time. Cooperative-thread strategies may be implemented using path-relinking, by combining elite solutions stored in a central pool with the local optima found by each processor at the end of each GRASP iteration. Canuto et al. [30] used path-relinking to implement a parallel GRASP for the prize-collecting Steiner tree problem. Their strategy is truly cooperative, since pairs of elite solutions from a centralized unique central pool are distributed to the processors which perform path-relinking in parallel. Computational results obtained with an MPI implementation running on a cluster of 32 400-MHz Pentium II processors showed linear speedups.

Aiex et al. [4] compared independent and cooperative parallel GRASP with path-relinking implementations for job shop scheduling. Both implementations used MPI on a SGI Challenge computer with 28 R10000 processors. The cooperative parallel GRASP maintains a pool of elite solutions at each processor. When a new incumbent (best so far) solution is found by any processor, that solution is sent to all other processors to be inserted into their respective elite set pools. Table 1 illustrates that while sub-linear speedups were observed for the independent parallel GRASP, super-linear speedups were achieved by the collaborative implementation.

6. Applications

GRASP was first applied to set covering [46] in 1989, and, since then, has been applied to a wide range of problem types. The reader is referred to Festa and Resende [54] and the URL `http://graspheuristic.org/annotated` for an extensive annotated bibliography of GRASP. We conclude this paper with a partial list of applications of GRASP, showing its wide applicability.

- routing [12, 15, 20, 32, 65];
- logic [38, 88, 93, 96];
- covering and partition [11, 13, 46, 56, 58];
- location [1, 37, 62, 110, 111];
- minimum Steiner tree [31, 76, 77, 78, 103];
- optimization in graphs [2, 48, 66, 86, 92, 98, 102, 52, 69, 35];
- assignment [45, 55, 70, 71, 79, 82, 87, 91, 89, 106, 60];
- timetabling, scheduling, and manufacturing [17, 18, 19, 24, 36, 40, 42, 43, 44, 49, 50, 64, 104, 105, 114, 7];
- transportation [12, 42, 45, 72, 21];
- power systems [25, 26, 16];
- telecommunications [2, 14, 62, 71, 91, 92, 100, 29, 83, 113, 63, 73];
- graph and map drawing [51, 68, 98, 102, 74, 75, 27];
- speech [34];
- statistics [80, 81];
- biology [10];
- mathematical programming [85];
- packing [33]; and
- VLSI [11], among other areas of application.

Referencias

[1] S. Abdinnour-Helm and S.W. Hadley. Tabu search based heuristics for multi-floor facility layout. *International Journal of Production Research*, 38:365–383, 2000.

[2] J. Abello, P.M. Pardalos, and M.G.C. Resende. On maximum clique problems in very large graphs. In J. Abello and J. Vitter, editors, *External Memory Algorithms and Visualization*, volume 50 of *DIMACS Series on Discrete Mathematics and Theoretical Computer Science*, pages 199–130. American Mathematical Society, 1999.

[3] R.K. Ahuja, J.B. Orlin, and D. Sharma. New neighborhood search structures for the capacitated minimum spanning tree problem. *Mathematical Programming*, 91:71–97, 2001.

[4] R.M. Aiex, S. Binato, and M.G.C. Resende. Parallel GRASP with path-relinking for job shop scheduling. *Parallel Computing*, ?(?):??–??, 2003.

[5] 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, 2000.

[6] R.M. Aiex, M.G.C. Resende, and C.C. Ribeiro. Probability distribution of solution time in GRASP: An experimental investigation. *Journal of Heuristics*, 8:343–373, 2002.

[7] M.S. Akturk and D. Ozdemir. A new dominance rule to minimize total weighted tardiness with unequal release dates. *European Journal of Operational Research*, 135:394–412, 2001.

[8] A.C. Alvim. Parallelization strategies for the metaheuristic GRASP. Master's thesis, Department of Computer Science, Catholic University of Rio de Janeiro, Brazil, 1998. In Portuguese.

[9] A.C. Alvim and C.C. Ribeiro. Load balancing for the parallelization of the GRASP metaheuristic. In *Proceedings of the X Brazilian Symposium on Computer Architecture*, pages 279–282, B´uzios, 1998. In Portuguese.

[10] A. Andreatta and C.C. Ribeiro. Heuristics for the phylogeny problem. *Journal of Heuristics*, 8:429–447, 2002.

[11] S. Areibi and A. Vannelli. A GRASP clustering technique for circuit partitioning. In J. Gu and P.M. Pardalos, editors, *Satisfiability Problems*, volume 35 of *DIMACS Series on Discrete Mathematics and Theoretical Computer Science*, pages 711–724. American Mathematical Society, 1997.

[12] M.F. Argüello, J.F. Bard, and G. Yu. A GRASP for aircraft routing in response to groundings and delays. *Journal of Combinatorial Optimization*, 1:211–228, 1997.

[13] M.F. Argüello, T.A. Feo, and O. Goldschmidt. Randomized methods for the number partitioning problem. *Computers and Operations Research*, 23:103–111, 1996.

[14] M. Armony, J.C. Klincewicz, H. Luss, and M.B. Rosenwein. Design of stacked self-healing rings using a genetic algorithm. *Journal of Heuristics*, 6:85–105, 2000.

[15] J.B. Atkinson. A greedy randomised search heuristic for time-constrained vehicle scheduling and the incorporation of a learning strategy. *Journal of the Operatinal Research Society*, 49:700–708, 1998.

[16] L. Bahiense, G.C. Oliveira, and M. Pereira. A mixed integer disjunctive model for transmission network expansion. *IEEE Transactions on Power Systems*, 16:560–565, 2001.

[17] J.F. Bard and T.A. Feo. Operations sequencing in discrete parts manufacturing. *Management Science*, 35:249–255, 1989.

[18] J.F. Bard and T.A. Feo. An algorithm for the manufacturing equipment selection problem. *IIE Transactions*, 23:83–92, 1991.

[19] J.F. Bard, T.A. Feo, and S. Holland. A GRASP for scheduling printed wiring board assembly. *IIE Transactions*, 28:155–165, 1996.

[20] J.F. Bard, L. Huang, P. Jaillet, and M. Dror. A decomposition approach to the inventory routing problem with satellite facilities. *Transportation Science*, 32:189–203, 1998.

[21] J.F. Bard, G. Kantoravdis, and G. Yu. A branch-and-cut procedure for the vehicle routing problem with time windows. *Transportation Science*, 36:250–269, 2002.

[22] R. Battiti and G. Tecchiolli. Parallel biased search for combinatorial optimization: Genetic algorithms and tabu. *Microprocessors and Microsystems*, 16:351–367, 1992.

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

[24] S. Binato, W.J. Hery, D. Loewenstern, and M.G.C. Resende. A GRASP for job shop scheduling. In C.C. Ribeiro and P. Hansen, editors, *Essays and Surveys in Metaheuristics*, pages 59–79. Kluwer Academic Publishers, 2002.

[25] S. Binato and G.C. Oliveira. A reactive GRASP for transmission network expansion planning. In C.C. Ribeiro and P. Hansen, editors, *Essays and Surveys in Metaheuristics*, pages 81–100. Kluwer Academic Publishers, 2002.

[26] S. Binato, G.C. Oliveira, and J.L. Ara´ujo. A greedy randomized adaptive search procedure for transmission expansion planning. *IEEE Transactions on Power Systems*, 16:247–253, 2001.

[27] C. Binucci, W. Didimo, G. Liotta, and M.Ñonato. Labeling heuristics for orthogonal drawings. In *Proceedings of GD'98 – Symposium on Graph Drawing*, volume 2265 of *Lecture Notes in Computer Science*, pages 139–153. Springer-Verlag, 2002.

[28] J.L. Bresina. Heuristic-biased stochastic sampling. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 271–278, Portland, 1996.

[29] M. Brunato and R. Battiti. A multistart randomized greedy algorithm for traffic grooming on mesh logical topologies. Technical report, Department of Mathematics, University of Trento, Trento, Italy, 2001.

[30] S.A. Canuto, M.G.C. Resende, and C.C. Ribeiro. Local search with perturbations for the prize-collecting Steiner tree problem in graphs. *Networks*, 38:50–58, 2001.

[31] S.A. Canuto, C.C. Ribeiro, and M.G.C. Resende. Local search with perturbations for

the prize-collecting Steiner tree problem. In *Extended Abstracts of the Third Metaheuristics International Conference*, pages 115–119, Angra dos Reis, July 1999.

[32] C. Carreto and B. Baker. A GRASP interactive approach to the vehicle routing problem with backhauls. In C.C. Ribeiro and P. Hansen, editors, *Essays and Surveys in Metaheuristics*, pages 185–199. Kluwer Academic Publishers, 2002.

[33] P. Chardaire, G.P. McKeown, and J.A. Maki. Application of GRASP to the multiconstraint knapsack problem. In E.J.W. Boers et al., editor, *EvoWorkshop 2001*, pages 30–39. Springer-Verlag Berlin Heidelberg, 2001.

[34] H. Fraçois and O. Boëffard. The greedy algorithm and its application to the construction of a continuous speech database. In *Proceedings of LREC-2002*, volume 5, pages 1420–1426, May 29-31 2002.

[35] A. Corber´an, R. Mart´ı, and J.M. Sanch´ıs. A GRASP heuristic for the mixed chinese postman problem. *European Journal of Operational Research*, 142:70–80, 2002.

[36] P. De, J.B. Ghosj, and C.E. Wells. Solving a generalized model for con due date assignment and sequencing. *International Journal of Production Economics*, 34:179–185, 1994.

[37] H. Delmaire, J.A. D´ıaz, E. Fern´andez, and M. Ortega. Reactive GRASP and Tabu Search based heuristics for the single source capacitated plant location problem. *INFOR*, 37:194–225, 1999.

[38] 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. *Mathematical Computer Modelling*, 27:75–99, 1998.

[39] N. Dodd. Slow annealing versus multiple fast annealing runs: An empirical investigation. *Parallel Computing*, 16:269–272, 1990.

[40] A. Drexl and F. Salewski. Distribution requirements and compactness constraints in school timetabling. *European Journal of Operational Research*, 102:193–214, 1997.

[41] H.T. Eikelder, M. Verhoeven, T. Vossen, and E. Aarts. A probabilistic analysis of local search. In I. Osman and J. Kelly, editors, *Metaheuristics: Theory and Applications*, pages 605–618. Kluwer Academic Publishers, 1996.

[42] T.A. Feo and J.F. Bard. Flight scheduling and maintenance base planning. *Management Science*, 35:1415–1432, 1989.

[43] T.A. Feo and J.F. Bard. The cutting path and tool selection problem in computer-aided process planning. *Journal of Manufacturing Systems*, 8:17–26, 1989.

[44] T.A. Feo, J.F. Bard, and S. Holland. Facility-wide planning and scheduling of printed wiring board assembly. *Operations Research*, 43:219–230, 1995.

[45] T.A. Feo and J.L. Gonz´alez-Velarde. The intermodal trailer assignment problem: Models, algorithms, and heuristics. *Transportation Science*, 29:330–341, 1995.

[46] 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.

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

[48] T.A. Feo, M.G.C. Resende, and S.H. Smith. A greedy randomized adaptive search procedure for maximum independent set. *Operations Research*, 42:860–878, 1994.

[49] T.A. Feo, K. Sarathy, and J. McGahan. A GRASP for single machine scheduling with sequence dependent setup costs and linear delay penalties. *Computers and Operations Research*, 23:881–895, 1996.

[50] T.A. Feo, K. Venkatraman, and J.F. Bard. A GRASP for a difficult single machine scheduling problem. *Computers and Operations Research*, 18:635–643, 1991.

[51] E. Fern´andez and R. Mart´ı. GRASP for seam drawing in mosaicking of aerial photographic maps. *Journal of Heuristics*, 5:181–197, 1999.

[52] P. Festa, P.M. Pardalos, and M.G.C. Resende. Algorithm 815: FORTRAN subroutines for computing approximate solution to feedback set problems using GRASP. *ACM Transactions on Mathematical Software*, 27:456–464, 2001.

[53] P. Festa, P.M. Pardalos, M.G.C. Resende, and C.C. Ribeiro. Randomized heuristics for the MAX-CUT problem. *Optimization Methods & Software*, 7(6):??–??, 2003.

[54] P. Festa and M.G.C. Resende. GRASP: An annotated bibliography. In C.C. Ribeiro and P. Hansen, editors, *Essays and Surveys in Metaheuristics*, pages 325–367. Kluwer Academic Publishers, 2002.

[55] 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.

[56] J.B. Ghosh. Computatinal aspects of the maximum diversity problem. *Operations Research Letters*, 19:175–181, 1996.

[57] F. Glover. Tabu search and adaptive memory programing – Advances, applications and challenges. In R.S. Barr, R.V. Helgason, and J.L. Kennington, editors, *Interfaces in Computer Science and Operations Research*, pages 1–75. Kluwer, 1996.

[58] P.L. Hammer and D.J. Rader, Jr. Maximally disjoint solutions of the set covering problem. *Journal of Heuristics*, 7:131–144, 2001.

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

[60] M. Hasan, I. Osman, and T. AlKhamis. A meta-heuristic procedure for the three dimension assignment problem. *International Journal of Applied Mathematics*, 8:365–380, 2002.

[61] H. Hoos and T. Stützle. Towards a characterisation of the behaviour of stochastic local search algorithms for SAT. *Artificial Intelligence*, 112:213–232, 1999.

[62] 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.

[63] J.G. Klincewicz. Enumeration and search procedures for a hub location problem with economies of scale. *Annals of Operations Research*, 110:107–122, 2002.

[64] J.G. Klincewicz and A. Rajan. Using GRASP to solve the component grouping problem. *Naval Research Logistics*, 41:893–912, 1994.

[65] G. Kontoravdis and J.F. Bard. A GRASP for the vehicle routing problem with time windows. *ORSA Journal on Computing*, 7:10–23, 1995.

[66] M. Laguna, T.A. Feo, and H.C. Elrod. A greedy randomized adaptive search procedure for the two-partition problem. *Operations Research*, 42:677–687, 1994.

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

[68] M. Laguna and R. Mart´ı. GRASP and path relinking for 2-layer straight line crossing minimization. *INFORMS Journal on Computing*, 11:44–52, 1999.

[69] M. Laguna and R. Mart´ı. A GRASP for coloring sparse graphs. *Computational Optimization and Applications*, 19:165–178, 2001.

[70] 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.

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

[72] H. Ramalhinho Lourenço, J.P. Paixão, and R. Portugal. Multiobjective metaheuristics for the bus-driver scheduling problem. *Transportation Sciences*, 35:331–343, 2001.

[73] P. Mahey and C.C. Ribeiro. Modeling modern multimedia traffic. *Annals of Operations Research*, 110:107–122, 2002.

[74] R. Martí. Arc crossing minimization in graphs with GRASP. *IIE Transactions*, 33:913–919, 2001.

[75] R. Martí and V. Estruch. Incremental bipartite drawing problem. *Computers and Operations Research*, 28:1287–1298, 2001.

[76] S.L. Martins, P.M. Pardalos, M.G.C. Resende, and C.C. Ribeiro. Greedy randomized adaptive search procedures for the steiner problem in graphs. In P.M. Pardalos, S. Rajasejaran, and J. Rolim, editors, *Randomization Methods in Algorithmic Design*, volume 43 of *DIMACS Series on Discrete Mathematics and Theoretical Computer Science*, pages 133–145. American Mathematical Society, 1999.

[77] S.L. Martins, M.G.C. Resende, C.C. Ribeiro, and P. Pardalos. A parallel GRASP for the Steiner tree problem in graphs using a hybrid local search strategy. *Journal of Global Optimization*, 17:267–283, 2000.

[78] 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 IRREGULAR'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.

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

[80] M.C. Medeiros, M.G.C. Resende, and A. Veiga. Piecewise linear time series estimation with GRASP. *Computational Optimization and Applications*, 19:127–144, 2001.

[81] M.C. Medeiros, A. Veiga, and M.G.C. Resende. A combinatorial approach to piecewise linear time analysis. *Journal of Computational and Graphical Statistics*, 11:236–258, 2002.

[82] 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, 1998.

[83] A. Myslek. Greedy randomised adaptive search procedures (GRASP) for topological design of mpls networks. In *Proceedings of the 8th Polish Teletraffic Symposium*, 2001.

[84] L. Osborne and B. Gillett. A comparison of two simulated annealing algorithms applied to the directed Steiner problem on networks. *ORSA Journal on Computing*, 3:213–225, 1991.

[85] G. Palubeckis and A. Tomkevicius. GRASP implementations for the unconstrained binary quadratic optimization problem. *Information Technology and Control*, 24:14–20, 2002.

[86] P. M. Pardalos, T. Qian, and M. G. C. Resende. A greedy randomized adaptive search procedure for the feedback vertex set problem. *Journal of Combinatorial Optimization*, 2:399–412, 1999.

[87] 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 115–133. Kluwer Academic Publishers, 1995.

[88] 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.

[89] L.S. Pitsoulis, P.M. Pardalos, and D.W. Hearn. Approximate solutions to the turbine balancing problem. *European Journal of Operational Research*, 130:147–155, 2001.

[90] L.S. Pitsoulis and M.G.C. Resende. Greedy randomized adaptive search procedures. In P.M. Pardalos and M.G.C. Resende, editors, *Handbook of Applied Optimization*, pages 168–183. Oxford University Press, 2002.

[91] 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.

[92] M.G.C. Resende. Computing approximate solutions of the maximum covering problem using GRASP. *Journal of Heuristics*, 4:161–171, 1998.

[93] 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.

[94] M.G.C. Resende, T.A. Feo, and S.H. Smith. Algorithm 787: Fortran subroutines for approximate solution of maximum independent set problems using GRASP. *ACM Trans. Math. Software*, 24:386–394, 1998.

[95] 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.

[96] 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.

[97] 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.

[98] M.G.C. Resende and C.C. Ribeiro. A GRASP for graph planarization. *Networks*, 29:173–189, 1997.

[99] M.G.C. Resende and C.C. Ribeiro. Greedy randomized adaptive search procedures. In F. Glover and G. Kochenberger, editors, *Handbook of Metaheuristics*, pages 219–249. Kluwer Academic Publishers, 2002.

[100] M.G.C. Resende and C.C. Ribeiro. A GRASP with path-relinking for private virtual circuit routing. *Networks*, 41(1):104–114, 2003.

[101] M.G.C. Resende and R.F. Werneck. A GRASP with path-relinking for the *p*-median problem. Technical report, Internet and Network Systems Research Center, AT&T Labs Research, Florham Park, NJ, 2002.

[102] C.C. Ribeiro and M.G.C. Resende. Algorithm 797: Fortran subroutines for approximate solution of graph planarization problems using GRASP. *ACM Transactions on Mathematical Software*, 25:342–352, 1999.

[103] C.C. Ribeiro, E. Uchoa, and R.F. Werneck. A hybrid GRASP with perturbations for the Steiner problem in graphs. *INFORMS Journal on Computing*, 14:228–246, 2002.

[104] R.Z. Ríos-Mercado and J.F. Bard. Heuristics for the flow line problem with setup costs. *European Journal of Operational Research*, pages 76–98, 1998.

[105] R.Z. Ríos-Mercado and J.F. Bard. An enhanced TSP-based heuristic for makespan minimization in a flow shop with setup costs. *Journal of Heuristics*, 5:57–74, 1999.

[106] A.J. Robertson. A set of greedy randomized adaptive local search procedure (GRASP) implementations for the multidimensional assignment problem. *Computational Optimization and Applications*, 19:145–164, 2001.

[107] B. Selman, H. Kautz, and B. Cohen. Noise strategies for improving local search. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 337–343, Seattle, 1994. MIT Press.

[108] D. Serra and R. Colomé. Consumer choice in competitive location models: Formulations and heuristics. *Papers in Regional Science*, 80:439–464, 2001.

[109] E. Taillard. Robust taboo search for the quadratic assignment problem. *Parallel Computing*, 7:443–455, 1991.

[110] T.L. Urban. Solution procedures for the dynamic facility layout problem. *Annals of Operations Research*, pages 323–342, 1998.

[111] T.L. Urban, W.-C. Chiang, and R.A. Russel. The integrated machine allocation and layout problem. *International Journal of Production Research*, pages 2913–2930, 2000.

[112] M.G.A. Verhoeven and E.H.L. Aarts. Parallel local search. *Journal of Heuristics*, 1:43–65, 1995.

[113] J. Xu and S. Chiu. Effective heuristic procedure for a field technician scheduling problem. *Journal of Heuristics*, 7:495–509, 2001.

[114] J. Yen, M. Carlsson, M. Chang, J.M. Garcia, and H.Ñguyen. Constraint solving for inkjet print mask design. *Journal of Imaging Science and Technology*, 44:391–397, 2000.