Ann Oper Res (2008) 159: 125–133
DOI 10.1007/s10479-007-0263-4

A GRASP algorithm for the multi-criteria minimum spanning tree problem

José Elias Claudio Arroyo · Pedro Sampaio Vieira · Dalessandro Soares Vianna

Published online: 30 November 2007

Abstract This paper proposes a GRASP (Greedy Randomized Adaptive Search Procedure) algorithm for the multi-criteria minimum spanning tree problem, which is NP-hard. In this problem a vector of costs is defined for each edge of the graph and the problem is to find all Pareto optimal or efficient spanning trees (solutions). The algorithm is based on the optimization of different weighted utility functions. In each iteration, a weight vector is defined and a solution is built using a greedy randomized constructive procedure. The found solution is submitted to a local search trying to improve the value of the weighted utility function. We use a *drop-and-add* neighborhood where the spanning trees are represented by Prufer numbers. In order to find a variety of efficient solutions, we use different weight vectors, which are distributed uniformly on the Pareto frontier.

The proposed algorithm is tested on problems with $r = 2$ and 3 criteria. For non-complete graphs with $n = 10$, 20 and 30 nodes, the performance of the algorithm is tested against a complete enumeration. For complete graphs with $n = 20$, 30 and 50 nodes the performance of the algorithm is tested using two types of weighted utility functions. The algorithm is also compared with the multi-criteria version of the Kruskal's algorithm, which generates supported efficient solutions.

Keywords GRASP algorithm · Multi-criteria combinatorial optimization · Minimum spanning tree

Many practical optimization problems, generally, involve the minimization (or maximization) of several conflicting decision criteria. For example, in the topological network design problem it is desirable to find the best layout of components optimizing performance criteria, such as financial cost, message delay, traffic, link reliability, and so on. These criteria

This work was funded by the Municipal Town Hall of Campos dos Goytacazes city. The used computer was acquired with resource of CNPq.

J.E.C. Arroyo · P.S. Vieira · D.S. Vianna (✉)
Universidade Candido Mendes—UCAM-Campos, Núcleo de Pesquisa e Desenvolvimento—NPD, Campos dos Goytacazes, Rio de Janeiro 28040-320, Brazil
e-mail: dalessandro@ucam-campos.br

are conflicting and cannot be optimized simultaneously. Instead, a satisfactory trade-off has to be found. So a *Decision Maker* has to select the best compromise solution, taking into account the preference of the criteria.

The goal of multi-criteria combinatorial optimization (MCCO) is to optimize simultaneously $r > 1$ criteria or objectives finding a satisfactory trade-off. MCCO problems have a set of optimal solutions (instead of a single optimum) in the sense that no other solutions are superior to them when all criteria are taken into account. They are known as *Pareto optimal* or *efficient solutions*.

Solving MCCO problems is quite different from single-objective case ($r = 1$), where an optimal solution is searched. The difficulty is not only due to the combinatorial complexity as in single-objective case, but also to the research of all elements of the efficient set, whose cardinality grows with the number of objectives.

In the literature, some authors have proposed exact methods for solving specific MCCO problems (Ehrgott and Gandibleux 2000). These methods are generally valid to bi-objective ($r = 2$) problems but can not be adapted easily to a higher number of objectives. Also, the exact methods are inefficient to solve large-scale NP-hard MCCO problems. As in the single-criterion case, the use of heuristic/metaheuristic techniques seems to be the most promising approach to MCCO because of their efficiency, generality and relative simplicity of implementation. These techniques generate good approximated solutions in a short computational time.

In a recent overview of multi-criteria metaheuristics, Jones et al. (2002) report the increase of papers published in the nineties and also note that almost 80% of the papers are dedicated to real problems, especially in the discipline of engineering. This number reflects not only the increasing awareness of problems with multiple criteria, but also that metaheuristics are effective techniques to cope with such problems. Metaheuristics such as genetic algorithms (Michalewicz 1996), tabu search (Glover et al. 1996) and simulated annealing (Kirkpatrick et al. 1983) were originally conceived for single-criterion combinatorial optimization and the success achieved in their application to a very large number of problems has stimulated researchers to extend them to MCCO problems. A striking revelation in the paper by Jones et al. (2002) is that 70% of the articles use genetic algorithms as the primary metaheuristic, 24% simulated annealing and 6% tabu search. In (Jones et al. 2002) is commented that, there is no sign in the literature reviewed of the newer metaheuristic techniques, such as GRASP, being applied in the multi-criteria case. More recently, Festa and Resende (2004) publish an annotated bibliography of the GRASP metaheuristic and there are not references of GRASP application to MCCO problems.

The common argument of researchers for their preference to genetic algorithms is that they naturally produce a population of solutions and therefore they are suitable for finding an approximation of the set of efficient solutions. This is not so trivial and a suitable technique to assign fitness to the elements of the population is crucial to evolve with a set of solutions homogeneously dispersed to the set of efficient solutions (Deb et al. 2000). Coello (2000) and Van Veldhuizen and Lamont (2000) present critical reviews and classification of the most important approaches to genetic algorithms for multi-criteria optimization.

In the multi-criteria minimum spanning tree (mc-MST) problem, a vector of real number (representing the cost of each criteria) is defined for each edge, and the problem is to find all Pareto optimal spanning trees. This problem is NP-hard (Ehrgott and Gandibleux 2000) and is not simply an extension of MST from single-criterion to multiple criteria. In general, we cannot get the optimal solution of the problem because these multiple criteria usually conflict with each other in practice. The mc-MST problem is commonly found in network-design oriented applications. For example, the edges costs may be associated with financial cost, message delay and traffic and link reliability.

The literature on mc-MST problem is rather scarce. An exact method is proposed in Ramos et al. (1998). In Ehrgott and Klamroth (1997) and Hamacher and Ruhe (1994) are proposed approximate polynomial algorithms. The method proposed in Knowles (2002) and Zhou and Gen (1999) are based on genetic algorithms.

In this paper we propose a GRASP algorithm to solve the mc-MST problem generating a set of approximated efficient solutions or *nondominated solutions*. In the construction phase is used the Kruskal's algorithm and in the local search procedure is used a *drop-and-add* neighborhood transformation. This drop-and-add operation uses Prufer numbers encoding (Moon 1967) for spanning tree representation.

The organization of this paper is as follows. In the next section, we present the formulation of a MCCO problem, give the uses concepts and present a formal definition of the mc-MST problem. In Sect. 2, we discuss with more details the proposed GRASP algorithm. We present computational results in Sect. 3. Finally, Sect. 4 contains our concluding remarks.

1 Multi-criteria optimization description

Let $G=(V,A)$ be a connected and undirected graph, where $V=\{v_1,\dots,v_n\}$ is a finite set of nodes and $A=\{e_1,\dots,e_m\}$ is a finite set of arcs or edges $e_k=(i,j)$, $i\in V$, $j\in V$, $i\neq j$. Each edges $e_k=(i,j)$ has associated a vector $c_{ij}=(c^1_{ij},\dots,c^r_{ij})$ of r positive real numbers (costs). A spanning tree of graph G is a subgraph $T=(V,A_T)$ with $A_T\subseteq A$, such that T contains all nodes in V and connects them with exactly $n-1$ edges, so that there are no cycles. The mc-MST problem can be formulated as:

$$\text{Minimize } f(T)=(f_1(T),\dots,f_r(T)) \quad \text{subject to} \quad T\in\Omega, \tag{1}$$

where $f_k(T)=\Sigma_{\forall(i,j)\in A_T} c^k_{ij}$ is the k-th objective function and Ω is the set of all the spanning trees of graph G. The image of a solution $T\in\Omega$ is the point $z=f(T)$ in the objective space $f(\Omega)$.

A point z *dominates* z' if $z_j=f_j(T)\le z'_j=f_j(T')'$, $\forall j=1,\dots,r$, and $z_j<z'_j$ for at least one j. A solution T dominates T' if $f(T)$ dominates $f(T')$. A solution $T^*\in\Omega$ is *Pareto optimal* (or *efficient*) if there is no $T\in\Omega$ such that $f(T)$ dominates $f(T^*)$. The goal is to determinate the set E of efficient solution. We call the representation of set E in $f(\Omega)$ as the *Pareto frontier*.

A utility function is a model of the Decision Maker's preferences that maps each point in the objective space into a value of utility. It assumed that the goal of the Decision Maker is to minimize the utility. We defined two types of utility functions.

Weighted Tchebycheff utility functions (u_∞) are defined in the following way:

$$u_\infty=\max_{j=1,\dots,r}\{\lambda_j(f_j(T)-z^*_j)\}, \tag{2}$$

where $\lambda=(\lambda_1,\dots,\lambda_r)$, $\forall j\lambda_j>0$, is a preference or weight vector and $z^*_j=\min_T\{z_j=f_j(T);\ j=1,\dots,r\}$. $z^*=(z^*_1,\dots,z^*_r)$ is called the ideal point. Each utility function of this type has at least one global optimum belonging to the set E. For each efficient solution T there is a weighted Tchebycheff utility function such that T is a global optimum of u_∞ (Steuer 1986).

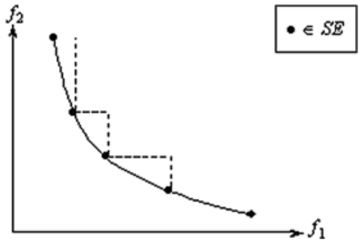

Fig. 1 Supported efficient solutions and the potential regions (*triangles*) of non-supported efficient solutions

Weighted linear utility functions (u_1) are defined in the following way:

$$u_1 = \sum_{j=1}^{r} \lambda_j f_j(T), \tag{3}$$

where $\Sigma_{j=1}^{r}\lambda_j = 1$ and $\lambda_j \geq 0$, $j = 1, \dots, r$.

In the convex case (the r functions f_j are convex and Ω is a convex set), the set E of efficient solution coincides with the set *SE* of the *supported efficient solutions*. An efficient solution T is supported if exists a weight vector $\lambda = (\lambda_1, \dots, \lambda_r)$ such that T is the unique global optimum of the following parameterized single objective problem:

$$\text{Minimize } \sum_{j=1}^{r} \lambda_j f_j(T) \quad \text{subject to} \quad T \in \Omega. \tag{4}$$

The efficient solutions which cannot be found by optimization of problem (4) are called *non-supported efficient solutions*. For two objectives case, these solution are necessarily located in the triangles generated in the objective space by two successive supported efficient solutions, as represented in Fig. 1.

2 The multi-criteria GRASP heuristic (mc-GRASP)

GRASP—Greedy Randomized Adaptive Search Procedure (Feo and Resende 1995) is a multi-start metaheuristic, in which each iteration consists of two phases: construction and local search. The construction phase builds a feasible solution using a greedy randomized algorithm, whose neighborhood is investigated until a local minimum is found during the local search phase. The best overall solution is kept as the result.

In the construction phase, we use the Kruskal's algorithm. The Kruskal's algorithm (Kruskal 1956) is a well-known polynomial time constructive algorithm for solving the simple criterion minimum spanning tree problem. Using data structures for disjoint sets, the Kruskal's algorithm can be implemented efficiently, requiring $O(m \log n)$ operations (Cormen 2001). A multi-criteria version of this algorithm, called, mc-Kruskal is easily devised. By simple replacing the vector of edges costs in the graph by a weighted sum scalarization of them, optimization can be carried out in one direction of the objective space. In mc-Kruskal this procedure is iterated for many different weight vectors, giving a whole range of solutions approximating the Pareto frontier.

2.1 mc-GRASP heuristic

The mc-GRASP heuristic is based on the optimization of a weighted utility function $u(T)$ corresponding to a solution T. The main idea of the heuristic is to define a weight vector for each iteration. The weight vector is used in the weighted utility function. In each iteration of the heuristic a solution T is built using the greedy randomized Kruskal's algorithm (see Sect. 2.2), and then, this solution is submitted to a local search procedure (see Sect. 2.3).

The weight vector $\lambda = (\lambda_1, \ldots, \lambda_r)$, generally, determinates a search direction on the Pareto optimal frontier and various search directions are required to find a variety of Pareto optimal solutions. Murata et al. (2001) introduces a way of generating weight vectors distributed uniformly on the Pareto frontier. The vectors are generated combining r non-negatives integers with the sum of s:

$$w_1 + w_2 + \cdots + w_r = s, \quad \text{where } w_i \in \{0, 1, 2, \ldots, s\}.$$

As an example, for $r = 2$ criteria and $s = 5$ we have 6 vectors: $(5,0)$, $(4,1)$, $(3,2)$, $(2,3)$, $(1,4)$ and $(0,5)$. For $r = 3$ and $s = 3$ we have 10 vectors: $(3,0,0)$, $(2,1,0)$, $(2,0,1)$, $(1,2,0)$, $(1,1,1)$, $(0,2,1)$, $(0,3,0)$, $(1,0,2)$, $(0,1,2)$ and $(0,0,3)$.

In order to obtain normalized weights ($\sum_{j=1}^{r} \lambda_j = 1$), we considered $\lambda_j = w_j/s$, $w_j \in \{0, \ldots, s\}$.

The number of generated vectors for r objectives and for a value of s, $N_r(s)$, is calculated as follows:

$$\begin{aligned}
N_2(s) &= s + 1, \\
N_3(s) &= \sum_{i=0}^{s} N_2(i) = (s+1)(s+2)/2, \\
N_4(s) &= \sum_{i=0}^{s} N_3(i) = \sum_{i=0}^{s} (i+1)(i+2)/2.
\end{aligned}$$

Algorithm 1 presents the implemented mc-GRASP algorithm that receives as input parameters the number of iterations *N_iter*, the percentage α ∈[0,1] (controls the amount of greediness and randomness) used at the construction phase and the weighted utility function to be optimized. As output, the algorithm returns the *lPareto* list, where the nondominated solutions are stored. The number of iterations of the algorithm corresponds to the number of weight vectors.

```
Algorithm 1 mc-GRASP($N\_iter$, $\alpha$, $u$)
01. $lPareto = \emptyset$;
02. Define a set of weight vectors $\Lambda = \{\lambda_i = (\lambda_1, \ldots, \lambda_r); i = 1, \ldots, N\_iter\}$;
03. For $i = 1$ to $N\_iter$ do
04.     $T$ = Greedy_Randomized_Kruskal($\alpha$, $\lambda^i$);
05.     Update_The_Pareto_List($T$, $lPareto$);
06.     Local_Search($T$, $\lambda^i$, $u(T)$, $lPareto$);
07. End-For
End-Algorithm
```

2.2 Greedy randomized construction

In the greedy randomized construction, for each edge (i, j) of the graph is computed the weighted sum $\lambda c_{ij} = \sum_{k=1}^{r} \lambda_k c_{ij}^k$, where $c_{ij} = (c_{ij}^1, \ldots, c_{ij}^r)$ is the cost vector of the edge (i, j) and $\lambda = (\lambda_1, \ldots, \lambda_r)$ is the weight vector.

The candidate list $C = \{e_1, \dots, e_m\}$ contains all the edges, in a no decreasing order of λc_{ij}. The restricted candidate list is defined as $RCL = \{e_1, \dots, e_{|RLC|}\}$, where $|RLC| = \mathtt{max}(1, \alpha \times |C|)$ is the cardinality of RLC and $\alpha \in [0,1]$. In each iteration of the constructive phase, an edge is selected randomly from RCL and it is added to the partial spanning tree as in the Kruskal's algorithm. This phase finalizes when the spanning tree has $n-1$ edges. The randomization is necessary to construct different initial solutions. In this way, it can be obtained efficient solutions that are not supported efficient solutions.

2.3 Local search

In the `Local_Search` procedure, a feasible spanning tree T is represented by a Prufer number P (vector with $n-2$ nodes) and by a permutation of the n nodes B. P and B are constructed using the Algorithm 2.

Algorithm 2 `Encode` (T)

```
01. $P = \emptyset$, $B = \emptyset$. All $n-1$ edges in the tree $T$ are labeled as temporary.
02. Construct a set $D_1 \subset V$ formed by degree 1 nodes of the temporarily
    labeled edges of $T$.
03. Choose node $k$, such that $k$ is the least index in $D_1$.
04. Consider edge $(k, j) \in T$. $B = B + k$ and $P = P + j$.
05. Give edge $(k, j) \in T$ a permanent label. If there is only a remaining edge
    $(u, v)$ with temporary label, add $u$ and $v$ to $B$, return $P$ and $B$, stop. Else,
go to Step 2.
```
End-Algorithm

The tree T in Fig. 2(a) is represented by $B = (b_j) = [1\ 3\ 4\ 2\ 6\ 5\ 7]$ and $P = (p_j) = [6\ 7\ 2\ 6\ 5]$. A new spanning tree T^* is formed by dropping and adding edges. To find a neighbor T^* of T, we select a index $j \in \{1, \dots, n-2\}$. The edge $e = (b_j, p_j) \in T$ is to be dropped. This deletion creates two sub-trees T_1 and T_2, rooted at b_j and p_j respectively. A new tree T^* is generated adding a edge $e^* = (b_j, k)$, $k \in T_2$, $k \neq p_j$. T^* is evaluated as follows: $f_i(T^*) = f_i(T) - c_e^i + c_{e^*}^i$, $i = 1, \dots, r$. If the new solution is accepted, we constructed the representations B^* and P^* of the tree T^* doing two passes trough B. In the first step, all the vertices $b_l \in T_1 \cap B - \{b_j\}$, are added to B^* and the correspondent adjacent vertices $p_l \in P$ to P^*. Next, the vertices in (b_j, k) are added to B^* and P^*, respectively, and finally all the vertices $b_l \in T_2 \cap B$ are added to B^* (and the corresponding vertices $p_l \in P$ to P^*, $l \leq n-2$). A neighbor T^* of T (Fig. 2(a)) is showed in Fig. 2(b). The dropped

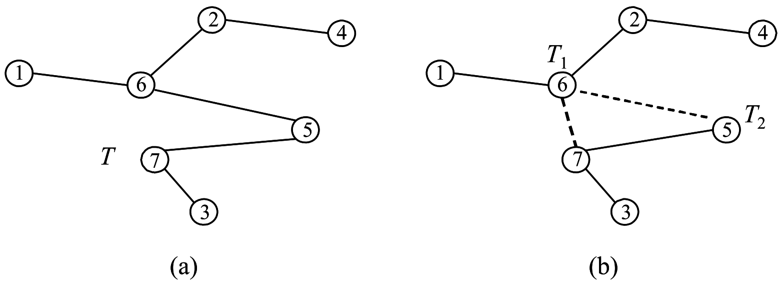

Fig. 2 (**a**) An example of an encoding, (**b**) drop-and-add neighborhood

edge is $(b_j, p_j) = (6, 5)$, $j = 5$, and the added edge is $(b_j, k) = (6, 7)$. The new tree T^* is represented by $B^* = [1\ 4\ 2\ 6\ 3\ 5\ 7]$ and $P^* = [6\ 2\ 6\ 7\ 7]$.

3 Computational experiments

All computational experiments were executed on a Pentium IV 2.2 GHz. The mc-GRASP algorithm was implemented in C. The goal of this work is to demonstrate that using a metaheuristic, such as GRASP, it can be obtained others efficient solutions, which can not be found with the mc-Kruskal algorithm. The mc-Kruskal algorithm generates some efficient solutions (or the supported efficient solutions).

The performance of mc-GRASP algorithm is tested on two sets of graphs with 2 and 3 criteria. The first set contains 5 graphs for each $(n, m) = (10, 45), (20, 30)$. For these graphs, the efficient solutions (for 2 and 3 criteria) are obtained by complete enumeration (EC). The second set contains 5 complete graphs for each $n = 20, 30, 50$ vertices and, in this case, the efficient solutions are not known. The edge costs c^1_{ij}, c^2_{ij} and c^3_{ij} are uniformly distributed in the intervals $[30, 200]$, $[20, 100]$ and $[10, 50]$, respectively (Zhou and Gen 1999). mc-GRASP was executed $N\_iter \approx 1000$ iterations. Best results were achieved for $\alpha = 0.1, 0.08, 0.03$ and 0.01 for graphs with $n = 10, 20, 30$ and 50, respectively.

We tested two types of weighted utility functions, linear and Tchebycheff. For mc-MST problems with two and three criteria, the weighted linear functions give better results.

In multi-criteria optimization, there is no natural single measure that is able to capture the quality of an approximation set H to the Pareto optimal set or reference set R (Arroyo and Armentano 2004). In this work we measure the performance of the nondominated set H generated by the heuristic method relative to the reference set R by using two measures:

- *Cardinal measure*: $NRS = |H \cap R|$ (number of reference solutions found by the heuristic method) and,
- *Distance measure*: $D_{av} = \frac{1}{|R|}\sum_{z\in R}\min_{z'\in H} d(z', z)$, where $d(z', z)$ is the Euclidean distance between z' and z. Note that D_{av} is the average distance from a point $z \in R$ to its closest point in H.

When the Pareto optimal set is not known and H' is the set of nondominated points generated by another heuristic, we define the reference set R as the nondominated points of $(H \cup H')$ (Arroyo and Armentano 2004) and use the same measures mentioned above to assess the approximation of H and H' relative to R.

The problems with $(n, m) = (10, 45)$ and $(20, 30)$ were solved to optimality by EC. For a total of 10 problems, 312 and 4011 efficient solutions were generated by the EC algorithm for 2 and 3 criteria, respectively. The mc-GRASP found 258 (82.70%) and 1949 (48.60%) efficient solutions for problems with 2 and 3 criteria, respectively. The mean computational time spend by the mc-GRASP on 10 problems with three criteria was 1.2 seconds.

The algorithms mc-GRASP and mc-Kruskal were tested on 15 complete graphs of the second set considering two and three criteria. For these problems the efficient solutions are not known. Therefore, their performance is evaluated with respect to a reference set R, which contains the nondominated solutions from all solutions generated by the two algorithms. Tables 1 and 2 display the results of the mc-GRASP algorithm and mc-Kruskal algorithm for two and three criteria, respectively. In these tables are showed the number of nondominated solutions in the set R and the number of reference solutions provided by each algorithm (*NRS*). The mc-Kruskal algorithm generates the supported efficient solutions.

Table 1 Results of `mc-GRASP` and `mc-Kruskal` on complete graphs with 2 criteria

Instance	$\lvert R\rvert$	mc-GRASP				mc-Kruskal		
		NS	*NRS*	D_{av}	Time	*NRS*	D_{av}	Time
$n = 20$	607	609	602	0.00001	0.25	138	0.00619	0.062
$n = 30$	1191	1192	1177	0.00002	0.47	244	0.00325	0.125
$n = 50$	2575	2579	2498	0.00002	1.20	461	0.00168	0.344
Total	**4373**	4380	**4277**			**843**		

Table 2 Results of `mc-GRASP` and `mc-Kruskal` on complete graphs with 3 criteria

Instance	$\lvert R\rvert$	mc-GRASP				mc-Kruskal		
		NS	*NRS*	D_{av}	Time	*NRS*	D_{av}	Time
$n = 20$	12437	12626	12123	0.00020	7.90	1213	0.01081	0.075
$n = 30$	27411	27800	26240	0.00020	21.84	2315	0.00773	0.156
$n = 50$	52192	52172	49345	0.00015	41.46	3882	0.00537	0.406
Total	**92040**	92601	**87708**			**7410**		

The `mc-GRASP` algorithm can generate supported efficient solutions and others efficient solutions, which can not be found by the `mc-Kruskal` algorithm. Tables 1 and 2 also display the computational time spent by the algorithms.

For the 15 problems tested with two criteria, 4373 reference solutions were generated, from which 4277 solutions were obtained by the `mc-GRASP` algorithm and 843 solutions by the `mc-Kruskal` algorithm.

For the 15 problems with three criteria, 92040 reference solutions were generated, from which 87708 solutions were obtained by the `mc-GRASP` algorithm and 7410 solutions by the `mc-Kruskal` algorithm. It is clear that the `mc-GRASP` algorithm generates a larger number of nondominated solutions for all problem sizes.

4 Conclusion

This paper presents a GRASP algorithm to generate a good approximation of the set of efficient solutions of the mc-MST problem. The algorithm is applied to solve problems with 2 and 3 criteria and the computational results show that the heuristic proposed obtained, for all instances tested, a number of reference solutions superior comparing with `mc-Kruskal` algorithm. It is also the most efficient algorithm when the quality of the obtained solutions (measured by the distance between solutions—D_{av}) is compared. High performance of the `mc-GRASP` algorithm is demonstrated by applying it to mc-MST problem.

It would be very interesting to apply the proposed algorithm to other types of combinatorial problems.

References

Arroyo, J. E. C., & Armentano, V. A. (2004). A partial enumeration heuristic for multi-objective flowshop scheduling problems. *Journal of Operations Research Society*, *55*, 1000–1007.

Coello, C. A. C. (2000). An updated survey of GA-based multiobjective optimization techniques. *ACM Computing Surveys*, *32*(2), 109–143.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2001). *Introduction to algorithms* (2nd ed.). McGraw-Hill: MIT Press.

Deb, K., Agrawal, S., Pratab, A., & Meyarivan, T. (2000). *A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II* (KanGAL Report 200001). Indian Institute of Technology, Kanpur, India.

Ehrgott, M., & Gandibleux, X. (2000). A survey and annotated bibliography of multiobjective combinatorial optimization. *OR Spektrum 22*, 425–460.

Ehrgott, M., & Klamroth, K. (1997). Connectedness of efficient solutions in multiple criteria combinatorial optimization. *European Journal of Operational Research*, *97*, 159–166.

Feo, T. A., & Resende, M. G. C. (1995). Greedy randomized adaptive search procedures. *Global Optimization 6*, 109–133.

Festa, P., & Resende, M. G. C. (2004). *An annotated bibliography of GRASP* (Technical Report). Submitted for the *European Journal of Operational Research*.

Glover, F. (1996). Tabu search and adaptive memory programming: Advances, applications and challenges. In: R. S. Barr, R. V. Helgason, & J. L. Kennington (Eds.), *Interfaces in computer science and operations research* (pp. 1–75). Kluwer Academic.

Hamacher, H. W., & Ruhe, G. (1994). On spanning tree problems with multiple objectives. *Annals of Operations Research*, *52*, 209–230.

Jones, D. F., Mirrazavi, S. K., & Tamiz, M. (2002). Multi-objective metaheuristics: An overview of the current state-of-art. *European Journal of Operational Research*, *137*, 1–19.

Kirkpatrick, S., Gellat Jr., C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, *220*, 671–680.

Knowles, J. D. (2002). *Local search and hybrid evolutionary algorithms for Pareto optimization*. Thesis of Doctorate, University of Reading, UK.

Kruskal, J. B. (1956). On the shortest spanning tree of graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, *7*, 48–50.

Michalewicz, Z. (1996). *Genetic algorithm + data structures = evolution programs*. Berlin: Springer.

Moon, J. W. (1967). Various proofs of Cayleys formula for counting trees. In Harary, F. (ed.), *A seminar on graph theory* (pp. 70–78). New York: Holt, Rinehart and Winston.

Murata, T., Ishibuchi, H., & Gen, M. (2001). Specification of genetic search directions in cellular multi-objective genetic algorithms. In *Lecture notes in computer science, Vol. 1993. Evolutionary multi-criterion optimization* (pp. 82–95). EMO. Zurich: Springer.

Ramos, R. M., Alonso, S., Sicília, J., & Gonzáles, C. (1998). The problem of the optimal biobjective spanning tree problem. *European Journal of Operational Research*, *111*, 617–628.

Steuer, R. E. (1986). *Multiple criteria optimization—theory, computation and application*. Wiley

Van Veldhuizen, D. A., & Lamont, G. B. (2000). Multiobjective evolutionary algorithms: Analyzing the state-of art. *Evolutionary Computation*, *8*(2), 125–147.

Zhou, G., & Gen, M. (1999). Genetic algorithm approach on multi-criteria minimum spanning tree problem. *European Journal of Operational Research*, *114*, 141–152.