CREATIVE MATH. & INF.
16 (2007), 42 - 53

Dedicated to Professor Ioan A. RUS on the occasion of his 70^{th} anniversary

Solving the generalized minimum spanning tree problem with simulated annealing

PETRICĂ POP, COSMIN SABO, CORINA POP SITAR and MARIAN V. CRĂCIUN

ABSTRACT. We consider a generalization of the minimum spanning tree problem, called the generalized minimum spanning tree problem, denoted by GMST. It is known that the GMST problem is $\mathcal{NP}$-hard. We present an effective algorithm for this problem. The method combines a simulated annealing algorithm (SA) with a local greedy algorithm. The heuristic that we proposed found solutions that were optimal for graphs with nodes up to 280 and were within at most 24% of optimality for larger problems, while the existing algorithms from the literature become computationally intractable.

1. INTRODUCTION

Many combinatorial optimization problems are $\mathcal{NP}$*-hard*, and the theory of $\mathcal{NP}$-completeness has reduced hopes that $\mathcal{NP}$*-hard* problems can be solved within polynomially bounded computation times. Nevertheless, sub-optimal solutions are sometimes easy to find. Consequently, there is much interest in approximation and heuristic algorithms that can find near optimal solutions within reasonable running time. Heuristic algorithms are typically among the best strategies in terms of efficiency and solution quality for problems of realistic size and complexity.

In contrast to individual heuristic algorithms that are designed to solve a specific problem, metaheuristics are strategic problem solving frameworks that can be adapted to solve a wide variety of problems. Metaheuristic algorithms are widely recognized as one of the most practical approaches for combinatorial optimization problems. Among representative metaheuristics are genetic algorithms, simulated annealing, tabu search and so on. Useful references regarding metaheuristic methods can be found in [7], [12] and [14].

The GMST problem was introduced in [11], and is defined on an undirected graph $G = (V, E)$ whose nodes are partitioned into a number of subsets (clusters) and whose edges have a nonnegative cost. The GMST problem asks for finding a minimum-cost tree in G that includes *exactly one* node from each cluster. Note that the well-known minimum spanning tree problem is a special case of the GMST problem where each cluster consists of only one node. The GMST problem has several applications to location and telecommunication problems.

The GMST problem was solved to optimality for nodes up to 200 by [3] using a branch-and-cut algorithm and by [13] using a rooting procedure based on a new mixed integer formulation of the GMST problem, but the problem becomes computationally intractable for larger instances.

Received: 20.09.2006. In revised form: 10.01.2007

2000 *Mathematics Subject Classification.* 90B10, 90C10, 90C27, 90C39.

Key words and phrases. *Generalized minimum spanning tree problem, dynamic programming, simulated annealing.*

More information on the problem and its applications can be found in [4], [11, 13].

The aim of this paper is to provide:

(i) an exact exponential time algorithm for the GMST problem, which has polynomial complexity when the number of clusters is fixed;

(ii) an effective heuristic based on simulated annealing.

The paper is organized as follows. In Section 2 we give the formal definition of the GMST problem and discuss its complexity. The exact algorithm is presented in Section 3. Two integer programming formulations of the GMST problem are presented in Section 4. In Section 5, we present an overview of simulated annealing. In Section 6, we solve the GMST problem with simulated annealing combined with a local greedy algorithm. In Section 7, we present the details of the heuristic implementation and the computational results obtained on various instances of the problem and compare them with previous results. Finally, concluding remarks are presented in Section 8.

2. Definition and complexity of the GMST problem

Let $G = (V, E)$ be an n-node undirected graph. Let $V_1, \ldots, V_m$ be a partition of V into m subsets called *clusters* (i.e., $V = V_1 \cup V_2 \cup \ldots \cup V_m$ and $V_l \cap V_k = \emptyset$ for all $l, k \in \{1, \ldots, m\}$ with $l \neq k$). We assume that edges are defined between all nodes which belong to different clusters. We denote the cost of an edge $e = (i, j) \in E$ by c_{ij} or by $c(i, j)$ and the costs of the edges are chosen integers.

The *generalized minimum spanning tree* (GMST) problem asks for finding a minimum-cost tree T spanning a subset of nodes which includes exactly one node from each cluster V_i, $i \in \{1, \ldots, m\}$. We will call such a tree a *generalized spanning tree*.

In [11] it is proved that the GMST problem is $\mathcal{NP}$-hard. [13] proved a stronger result:

Theorem 2.1. *The Generalized Minimum Spanning Tree problem on trees is $\mathcal{NP}$-hard.*

The proof of this result is based on a polynomial reduction of the set covering problem, which is known to be $\mathcal{NP}$-hard (see for example [5]), to the GMST problem defined on trees.

3. An Exact Algorithm for the GMST Problem

In this section, we present an algorithm that finds an exact solution to the GMST problem based on dynamic programming (see also [13]).

Let G' be the graph obtained from G after replacing all nodes of a cluster V_i with a supernode representing V_i. For convenience, we identify V_i with the supernode representing it. Edges of the graph G' are defined between each pair of the graph vertices $\{V_1, ..., V_m\}$.

Given a spanning tree of G', which we shall refer to as a *global spanning tree*, we use dynamic programming in order to find the corresponding best (w.r.t. cost minimization) generalized spanning tree.

Fix an arbitrary cluster V_{root} as the root of the global spanning tree and orient all the edges away from vertices of V_{root} according to the global spanning tree. A directed edge $\langle V_k, V_l \rangle$ of G', resulting from the orientation of edges of the global spanning tree defines naturally an orientation $\langle i, j \rangle$ of an edge $(i, j) \in E$ where $i \in V_k$ and $j \in V_l$. Let v be a vertex of cluster V_k for some $1 \leq k \leq m$. All such nodes v are potential candidates to be incident to an edge of the global spanning tree. On the graph G, we denote by $T(v)$ denote the subtree rooted at such a vertex v from G; $T(v)$ includes all vertices reachable from v under the above orientation of the edges of G based on the orientation of the edges of the global spanning tree. The *children* of $v \in V_k$, denoted by $C(v)$, are those vertices $u \in V_l$ which are heads of the directed edges $\langle v, u \rangle$ in the orientation. The leaves of the tree are those vertices that have no children.

Let $W(T(v))$ denote the minimum weight of a generalized subtree rooted at v. We want to compute

$$\min_{r \in V_{root}} W(T(r)).$$

We are now ready for giving the dynamic programming recursion to solve the subproblem $W(T(v))$. The initialization is:

$$W(T(v)) = 0, \text{ if } v \in V_k \text{ and } V_k \text{ is a leaf of the global spanning tree.}$$

To compute $W(T(v))$ for an interior to a cluster vertex $v \in V$, i.e., to find the optimal solution of the subproblem $W(T(v))$, we have to look at all vertices from the clusters V_l such that $C(v) \cap V_l \neq \emptyset$. If u denotes a child of the interior vertex v, then the recursion for v is as follows:

$$W(T(v)) = \sum_{l, C(v) \cap V_l \neq \emptyset} \min_{u \in V_l} [c(v, u) + W(T(u))].$$

Hence, for fixed v we have to check at most n vertices. Consequently, for the given global spanning tree, the overall complexity of this dynamic programming algorithm is $O(n^2)$. Since by Cayley's formula (see [1]), the number of all distinct global spanning trees is m^{m-2}, we have established the following.

Theorem 3.2. *There exists a dynamic programming algorithm which provides an exact solution to the GMST problem in $O(m^{m-2}n^2)$ time, where n is the number of nodes and m is the number of clusters in the input graph.*

Clearly, the above is an exponential time algorithm unless the number of clusters m is fixed.

4. Integer Programming Formulations

The GMST problem can be formulated as an integer program in many different ways, cf. [3], [4], [11] and [13].

For example, introducing the variables $x_e \in \{0, 1\}$, $e \in E$ and $z_i \in \{0, 1\}$, $i \in V$, to indicate whether an edge e respectively a node i is contained in the spanning tree, a feasible solution to the GMST problem can be seen as a cycle free subgraph

with $m-1$ edges, one node selected from every cluster and connecting all the clusters. Therefore the GMST problem can be formulated as the following 0-1 integer programming problem:

$$\begin{aligned}
\min \quad & \sum_{e \in E} c_e x_e \\
s.t. \quad & z(V_k) = 1, && \forall\, k \in K = \{1, ..., m\} && (4.1)\\
& x(E(S)) \leq z(S - i), \ \forall\, i \in S \subset V, 2 \leq |S| \leq n-1 && (4.2)\\
& x(E) = m - 1 && (4.3)\\
& x_e \in \{0, 1\}, && \forall\, e \in E && (4.4)\\
& z_i \in \{0, 1\}, && \forall\, i \in V. && (4.5)
\end{aligned}$$

Here $E(S) = \{e = (i, j) \ : \ i, j \in S\}, S \subseteq V$ we use the standard shorthand notations:

$$x(F) = \sum_{e \in F} x_e, \ F \subseteq E \text{ and } z(S) = \sum_{i \in S} z_i, \ S \subseteq V$$

For simplicity we used the notation $S - i$ instead of $S \setminus \{i\}$. In the above formulation, constraints (1) guarantee that from every cluster we select exactly one node, constraints (2) eliminate all the subtours and finally constraint (3) guarantees that the selected subgraph has $m - 1$ edges.

This formulation, introduced by [11], is called the *generalized subtour elimination formulation* since constraints (2) eliminate all the cycles.

We may replace the subtour elimination constraints (2) by connectivity constraints, resulting in the so-called *generalized cutset formulation* introduced by [11]:

$$\begin{aligned}
\min \quad & \sum_{e \in E} c_e x_e \\
s.t. \quad & (1), (3), (4), (5) \text{ and} \\
& x(\delta(S)) \geq z_i + z_j - 1, \ \forall\, i \in S \subset V, j \notin S && (4.6)
\end{aligned}$$

where for $S \subseteq V$, the *cutset*, denoted by $\delta(S)$, is defined as usually:

$$\delta(S) = \{\ e = (i, j) \in E \mid i \in S, \ j \notin S\}$$

We observe that the two formulations described have an exponential number of constraints since we have to choose subsets of V (constraints (2) and (6)).

5. Simulated Annealing

Simulated annealing (SA) is a global optimization method that distinguishes between different local optima and is inspired from Monte Carlo methods in statistical mechanics. The ideas that form the basis of simulated annealing were first published by [10] as an algorithm to simulate the cooling of material in a heat bath, a process known as annealing.

[9] and [2] suggested that this type of simulation could be used to search the feasible solutions of an optimization problem. Their approach can be regarded as a variant of the local search technique.

A central problem for local search is the occurrence of *local optima*, i.e. nodes in the search space where no neighbor strictly improves over the current node in terms of the cost function, but which are not global optima. In order to overcome the local optima, in the simulated annealing heuristic non-improving local moves are allowed, but their frequency is governed by a probability function which changes as the algorithm progresses. It has been shown (see [6]) that with a "large enough" initial temperature and a proper cooling schedule, SA guarantees a globally optimal solution. However, this theoretical result is particularly helpful, since the annealing time to ensure a significant probability of success will usually exceed the time required for a complete search of the solution space.

The annealing algorithm for a minimization problem with solution space S, objective function c and neighborhood structure N can be stated as follows:

Select a starting solution $s_0 \in S$, an initial temperature $T_0 > 0$, a temperature reduction function α and the number of iterations per temperature L;
Repeat
  Repeat
    Randomly select $s \in N(s_0)$ and compute $\delta = c(s) - c(s_0)$;
    If $\delta < 0$
    then $s_0 = s$
    else generate random p uniformly distributed in the range $(0, 1)$;
    if $p < \exp(-\delta/T)$ then $s_0 = s$
  Until $iterationcount = L$;
Set $T = \alpha(T)$;
Until stopping condition is true.
s_0 is the approximation of the optimal solution.

Here L represents the number of repetitions at each temperature, i.e. the number of randomly chosen candidate solutions in the neighborhood of the current solution that are evaluated. Therefore at each stage, L randomly chosen candidate solutions in the neighborhood of the current solution are evaluated. If a candidate solution improves on the current solution, it is accepted. Otherwise, it is accepted with a probability $p(T, \delta) = \exp(-\delta/T)$, which depends on the control parameter T (the temperature in the physical equivalent) and the amount δ by which a move worsens the current solution. This relation ensures that the probability of accepting a move to a very poor solution is very small. At the completion of each stage, the temperature is reduced. The way that the temperature is reduced is known as *cooling schedule*. Given a relatively high temperature T at the beginning of the process, the probability of accepting non-improving moves is fairly high. As the process continues, the temperature decreases and non-improving moves become less likely.

The search is continued until there is some evidence to suggest that there is a very low probability of improving on the best solution found so far. At this stage the system is said to be frozen.

6. SOLVING THE GMST PROBLEM WITH SIMULATED ANNEALING

In order to implement the simulated annealing algorithm for the GMST problem a number of decisions must be made. These can be divided into two categories: *generic decisions*, which are concerned with parameters of the annealing algorithm itself; including factors such as the initial temperature, the cooling schedule (governed by the parameter L, i.e. number of repetitions and the choice of the temperature reduction function α) and the stopping condition and *problem specific decisions*, which involves the choice of the solution space, the form of the cost function, an initial solution (initialization) and the neighborhood structure employed.

In our implementation of the simulated annealing algorithm for the GMST problem we chose the following cooling schedule:

1. **Initial value of the control parameter,** T_0.
   Following [8], we determine T_0 by calculating the average increase in cost δ^+, for a number of random transitions:
   $$T_0 = \frac{\delta^+}{\ln(\chi_0^{-1})}$$
   where $\chi_0 = 0.8$ is a given acceptance rate.
2. **Final value of the control parameter**.
   The stopping criteria, determining the final value of the control parameter, that we used was by fixing the number of values T_k, for which the algorithm is to be executed, ending for a suitable low temperature.
3. **Number of repetitions at each temperature**.
   The number of iterations at each temperature which is related to the size of the neighborhoods may vary from temperature to temperature. It is important to spend a long time at lower temperatures to ensure that a local optimum has been fully explored. Therefore we increased the value of L arithmetically (by adding a constant factor).
4. **Decrement of the control parameter**.
   We used the following decrement rule:
   $$\alpha(T) = rT$$
   where r is a constant smaller than but close to 1, called the cooling rate. We used, as [8], $r = 0.95$. This corresponds to a fairly slow cooling.

In what follows we present the problem specific decisions.

6.1. **Solution space.** We define the solution space as the set of all the feasible solutions. Suppose that the graph $G = (V, E)$ consists of n nodes which are partitioned into m clusters $V_1, ..., V_m$ having the same number of nodes, i.e.

$$|V_1| = ... = |V_m| = \frac{n}{m}$$

By Cayley's formula we know that the total number of global spanning trees (i.e. trees connecting the clusters) is equal to m^{m-2}. Given a global spanning tree there are $(\frac{n}{m})^m$ possible generalized spanning trees (feasible solutions of the GMST problem).

Therefore the solution space of the GMST problem contains

$$m^{m-2} \cdot (\frac{n}{m})^m = n^m \cdot m^{-2}$$

elements. The number of elements in the solution space is exponentially large, so that they cannot be explored exhaustively.

6.2. **Objective function.** We will consider the objective function described in the two integer programming formulations of the GMST problem, see Section 4.

6.3. **Initialization.** Simulated Annealing is an improvement heuristic which requires an initial solution. It is desirable to start with a relatively good solution (as compared, for example with a randomly generated one), otherwise a great deal of effort will be spent to simply reach the first local minimum.

In our case, an initial feasible solution for the GMST problem was found using the following greedy algorithm:

- **Input:** A connected graph $G = (V, E)$ with the nodes partitioned into m clusters and edges defined between nodes from different clusters with positive cost.
- **Output:** A generalized spanning tree $T = (W, F)$ of G.
  $F := \emptyset$ (F is the edge set of the current tree T)
  $W := \{v_i\}$, $v_i \in V_k$, $1 \leq k \leq m$ (W is the node set of T)
  while $|S| < m$ **do**

  Among all the edges with exactly one end in W find an edge (v_i, v_j) with minimum cost and whenever we choose a node from a cluster delete all the other nodes from that cluster and all the edges adjacent to the deleted nodes;
  $F := F \cup (v_i, v_j)$
  $W := W \cup (\{v_i, v_j\} \setminus W)$
  end; (while).

A similar algorithm as the greedy algorithm can be applied in order to find a feasible solution for the GMST problem (i.e. a generalized spanning tree of G) if we start with a generalized tree $T = (W, F)$ of G ($|W| < m$), instead of selecting randomly a node in a random cluster. We will call this algorithm the *local greedy algorithm*.

6.4. **Neighborhood structure.** The neighborhood is one of the most important factors in designing a simulated annealing algorithm. It should be designed such that the structures common to good solutions are preserved after the neighborhood operation is applied.

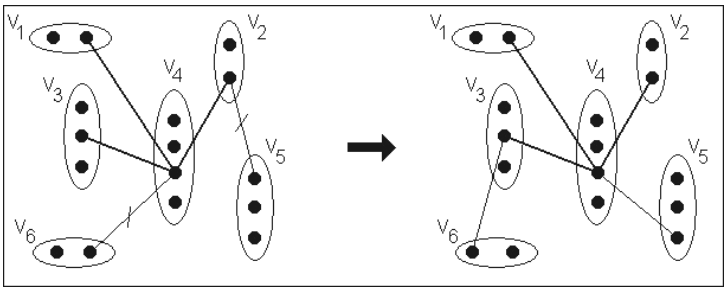

FIGURE 1. A current solution and a modified solution of the GMST problem

In the case of the GMST problem, we defined a neighborhood structure as follows: The *neighborhood* of any given generalized spanning tree T is defined as the set of all the generalized spanning trees obtained using the local greedy algorithm for any subtree of T. The size of the neighborhood of any given generalized spanning tree T is given by the number of subtrees. The maximum number of subtrees of a tree with m nodes is obtained for trees with the nodes distributed on exactly one level except the root which has degree $m-1$. Therefore in this case, the maximum size of the neighborhood is $2^{m-1}+m-1$, see [15]. Given a current solution of the GMST problem, i.e. a generalized spanning tree T, for any subtree U of T, using the local greedy algorithm we obtain a candidate solution for the simulated annealing in the neighborhood of the current solution. A current solution and a modified solution are shown in the next figure.

In Figure 1, we consider a graph G whose nodes are partitioned into 6 clusters and a current solution T of the problem. Selecting randomly a subtree U of T (for example discarding the edges connecting nodes from the clusters V_4 with V_5 and V_2 with V_5) we apply the local greedy algorithm in order to obtain the best modified solution in the neighborhood of T given the subtree U.

7. IMPLEMENTATION DETAILS AND COMPUTATIONAL RESULTS

Our computational experiments were performed on a Dual Intel Pentium III Processor - 1000 Mhz computer with 512 MB RAM. The combined Simulated Annealing and local greedy scheme procedures were written in Java.

After some preliminary tests, we have chosen the following cooling schedule: we start with a temperature $T_0 = 100$, the cooling rate $r = 0.95$ will be used to reduce the temperature in each stage and $L = 30$ moves are evaluated at each stage.

We considered an undirected graph $G = (V, E)$ having n nodes which are partitioned into m clusters such that each cluster contains the same number of nodes. Edges are defined between all the nodes from different clusters and the cost of the edges were generated randomly in the interval [1,100]. For each type of instance (n, m) we generated randomly five different problems, i.e. with different costs of the edges. The instances that we used are similar to those considered by Myung [11] and Feremans [3].

In the next table we compare the computational results obtained for solving the GMST problem using the combined simulated annealing with a local greedy algorithm with the computational results given by [3] and [13].

Table 1: Computational results for solving the GMST problem

Pb. size		*SA algorithm*		*Rooting procedure*		*Branch and cut*	
m	n	UB/OPT	CPU	LB/OPT	CPU	LB/UB	CPU
8	24	100	2.6	100	0.0	100	0.0
	32	100	3.7	100	0.0	100	0.2
	48	100	5.6	100	0.3	100	1.4
	80	100	10.2	100	2.2	100	4.2
10	30	100	4.8	100	0.2	100	1.0
	40	100	6.7	100	0.3	100	1.0
	60	100	10.8	100	0.7	100	3.2
	100	100	20.6	100	1.9	100	8.8
12	36	100	8.3	100	0.7	100	1.8
	48	100	11.6	100	0.9	99.2	3.2
	72	100	19.1	100	3.7	100	6.8
	120	100	39.0	100	9.3	-	-
15	45	100	16.6	100	1.3	100	3.6
	60	100	21.6	100	1.7	97.1	6.2
	90	100	39.7	100	9.6	100	21.4
	150	100	80.2	100	34.1	98.8	42.4
18	54	100	27.2	100	2.6	99.5	7.6
	108	100	68.1	100	13.8	-	-
	180	100	143.8	100	150.4	-	-
20	60	100	37.5	100	7.2	-	-
	120	100	181.9	100	33.2	96.3	39.8
	200	100	219.8	100	203.8	94.6	191.4
25	75	100	80.8	100	17.5	-	-
	150	100	221.7	100	266.4	88.3	178.8
	200	100	331.1	100	284.5	97.8	140.6
30	90	100	150.4	100	60.2	-	-
	180	100	412.0	100	721.8	96.6	114.6
	240	100	627.4	100	1948.9	-	-
40	120	100	382.1	100	142.6	100	92.6
	160	100	561.0	100	572.3	94.2	288.6
	280	100	691.2	100	25672.3	-	-

The first two columns in the table give the size of the problem: the number of clusters (m) and the number of nodes (n). The next two columns describe the simulated annealing procedure and contain: the upper bounds obtained as a percentage of the optimal value of the GMST problem (UB/OPT) and the computational

times (CPU) in seconds for solving the GMST problem. The last columns contain the lower bounds as a percentage of the optimal value of the GMST problem (UB/OPT) and the computational times (CPU) in seconds for solving the GMST problem with the rooting procedure (see [13]) and the lower bounds as a percentage of the upper bound of the GMST problem (LB/UB) and the computational times (CPU) in seconds for solving the GMST problem with a branch and cut algorithm (see [3]). In the last two columns '-' means that for those instances, [3] did not provide computational results.

It is important to notice that for all the instances considered we were able to obtain the optimal solution of the GMST problem using our simulated annealing combined with a local greedy algorithm.

From Table 1, we observe that the computational times in the case of simulated annealing were bigger compared with the computational times corresponding to the other methods for instances up to 140 nodes, but for larger instances, while the computational times for the rooting procedure increase exponentially, the computational times in the case of simulated annealing are reasonable.

For graphs which contains more than 50 clusters and 200 nodes, we obtained good solutions which are within at most 24 $\%$ of optimality. The computational results obtained in this case are presented in Table 2. We have considered again for each type of instance five trials.

Table 2: Computational results for solving large instances of the GMST problem (average of five trials per type of instance)

Pb. size		*Simulated Annealing algorithm*				
m	n	LB	UB	Average gap(%)	initial sol.	CPU
50	200	49	61	24	71	765.7
	300	49	55	12	67	1620.5
75	300	74	97	10	104	3545.0
	450	74	75	1	87	7440.3
100	300	99	114	15	128	6734.5
	400	99	103	4	118	10940.3
	600	99	99	0	107	20154.7

The first two columns in the table give the size of the problem: the number of clusters (m) and the number of nodes (n). The next four columns describe the simulated annealing procedure and contain: the lower bounds (the optimum for a graph whose nodes are partitioned into m clusters is at least $m-1$), the upper bounds obtained (UB), the average gap as a percentage, initial solutions and the

computational times (CPU) in seconds for calculating the upper bound of GMST problem.

As Table 2 shows, for large instances of the GMST problem our simulated algorithm provided good sub-optimal solutions within reasonable running time, while the other algorithms become computationally intractable.

The explanation for the improvement of the quality solution as the number of nodes increases for a fixed number of clusters is the following: when the number of nodes per cluster is increased, then the number of edges is increased and therefore the number of edges with low cost.

Analyzing the results presented in Tables 1 and 2, we observe that the number of clusters is a more important factor than the number of nodes for the behavior of the algorithms.

The quality of the solutions obtained by using the combined simulated annealing with local greedy algorithm depended on the initial solutions. In about 40% of the instances considered we were able to get the optimal solution using the first initial solution. We observed also that good initial solutions were not necessarily improved to good solutions by simulated annealing.

8. Conclusions

In this paper we have provided an exact exponential time algorithm for the generalized minimum spanning tree problem and shown an effective way of solving the GMST problem using simulated annealing combined with a local greedy algorithm. Our method has been tested and compared with previous methods from literature on various small to large instances of the problem that are commonly used for comparative studies. For instances up to 280 nodes our algorithm provides the optimal solution of the GMST problem and for large instances provides good sub-optimal solutions within reasonable running time.

References

[1] Bondy A., Thomassé S. and Thomasen C., *Spanning trees of multipartite graphs,* Preprint, 2001

[2] Cerny V., *A thermodynamical approach to the traveling salesman problem: an efficient simulated annealing algorithm,* Journal of Optimization Theory and Applications 45 (1985) 41-55

[3] Feremans C., *Generalized Spanning Trees and Extensions*, PhD Thesis, Universite Libré de Bruxelles, Belgium, 2001

[4] Feremans C., Labbé M. and Laporte G., *A Comparative Analysis of Several Formulations for the Generalized Minimum Spanning Tree Problem,* Technical Report IS-MG 2000/22, Universite Libré de Bruxelles, Belgium; to appear in Networks

[5] Garey M.R. and Johnson D.S., *Computers and Intractability: A guide to the theory of NP-Completeness, Freeman,* San Francisco, California, 1979

[6] Geman S. and Geman D., *Stochastic Relaxation, Gibbs Distribution and the Bayesian Restoration in Images*, IEEE Trans. Patt. Anal. Mac. Int. 6, 721-741

[7] Glover F.W. and Kochenberger G.A., Handbook of Metaheuristics, Kluwer Academic Publishers, 2002

[8] Johnson D., Aragon C., McGeoch L. and Schevon C., *Optimization by Simulated Annealing: An Experimental Evaluation,* Part I, Graph Partitioning, Operations Research 37 (1989), 865-892

[9] Kirkpatrick S., Gelatt C.D. and Vecchi M.D. *Optimization by Simulated Annealing*, Science, Vol. 220, No. 4598 (1983), 671-680

[10] Metropolis N., Rosenbluth A.W., Rosenbluth M.N., Teller A.H. and Teller E., *Equation of State Calculations by Fast Computing Machines*, Journal of Chem. Phys., Vol. 21, No. 6 (1953), 1087-1092
[11] Myung Y.S., Lee C.H. and D.w. Tcha, *On the Generalized Minimum Spanning Tree Problem*, Networks 26, pp. 231-241, 1995
[12] Osman I.H. and Laporte G., Metaheuristics, A Bibliography, Annals of Operations Research, 63, pp. 231-241, 1996
[13] Pop P.C., *The Generalized Minimum Spanning Tree Problem*, PhD thesis, University of Twente, The Netherlands, 2002
[14] Reeves C.R., *Modern Metaheuristic Techniques for Combinatorial Problems*, Blackwell, Oxford, 1993
[15] Szekely L.A. and Wang H., *On subtrees of trees*, Research report, University of Carolina, 2004

PETRICĂ POP
NORTH UNIVERSITY OF BAIA MARE
DEPARTMENT OF MATHEMATIS AND COMPUTER SCIENCE
VICTORIEI 76, 430122 BAIA MARE, ROMANIA
E-mail address: pop_petrica@yahoo.com

CORINA POP SITAR
NORTH UNIVERSITY OF BAIA MARE
DEPARTMENT OF ECONOMICS
VICTORIEI 76, 430122 BAIA MARE, ROMANIA
E-mail address: sitarcorina@yahoo.com

COSMIN SABO
NORTH UNIVERSITY OF BAIA MARE
DEPARTMENT OF MATHEMATISC AND COMPUTER SCIENCE
VICTORIEI 76, 430122 BAIA MARE, ROMANIA
E-mail address: cosmin_sabo@scream.ro

MARIAN V. CRĂCIUN
UNIVERSITY "DUNĂREA DE JOS" GALAŢI
DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE
DOMNEASCĂ 47, 800008 GALAŢI, ROMANIA
E-mail address: mcraciun@ugal.ro