MATHEMATICA, Tome 48 (71), N° 2, 2006, pp. 191–201

A LAGRANGIAN RELAXATION APPROACH TO THE GENERALIZED MINIMUM SPANNING TREE PROBLEM

PETRICĂ C. POP

Abstract. The Generalized Minimum Spanning Tree Problem, denoted GMST, is a variant of the classical Minimum Spanning Tree problem, and consists of finding a minimum-cost tree spanning a subset of nodes which includes exactly one node from every cluster in an undirected graph whose nodes are partitioned into clusters and whose edges are defined between nodes belonging to different clusters. The GMST problem is $\mathcal{N}P$-*hard* even when defined on trees. In this paper we consider an approach based on Lagrangian relaxation of the bidirectional flow formulation of the GMST problem. The subgradient method is used to obtained lower bounds. Computational results are reported for many instances of the problem.

MSC 2000. 05C05, 68R10, 90C05, 90C10, 90C27, 90C39.

Key words. Minimum spanning tree, generalized minimum spanning trees, $\mathcal{N}P$-*hard*, Lagrangian relaxation, subgradient method.

1. INTRODUCTION

We consider the generalized version of the minimum spanning tree problem (MST) called the generalized minimum spanning tree problem (GMST). Given an undirected graph whose nodes are partitioned into a number of subsets (clusters), the GMST problem is then to find a minimum-cost tree which includes *exactly* one node from each cluster. Therefore, the MST is a special case of the GMST problem where each cluster consists of exactly one node.

The GMST problem has been introduced by Myung, Lee and Tcha in [7] and the same authors showed that the problem is $\mathcal{N}P$-*hard*. A stronger result regarding its complexity has been provided by Pop [9] namely, the GMST problem even defined on trees is $\mathcal{N}P$-*hard*. The GMST problem has several applications to location problems, telecommunications, (see [6] and [11]), network design, railway optimization etc.

Myung *et al.* in [7] used a branch and bound procedure in order to solve the GMST problem. Their lower procedure is a heuristic method which approximates the linear programming relaxation associated with the dual of the multicommodity flow formulation of the GMST problem. They developed also a heuristic algorithm which finds a primal feasible solution for the GMST problem using the obtained dual solution. The GMST problem was solved to optimality for nodes up to 200 by Feremans [3] using a branch-and-cut algorithm. Pop [10] solved the problem to optimality for nodes up to 240 using a rooting procedure based on a local-global formulation of the problem. More information on the problem can be found on Feremans [3], Feremans *et al.* [4], Myung, Lee and Tcha [7] and Pop [8, 9, 10] .

A variant of the GMST problem is the problem of finding a minimum cost tree including *at least* one vertex from each cluster. This problem was introduced by Dror *et al.* in [2]. These authors provide also five heuristics including a genetic algorithm. In the present paper we confine ourselves to the problem of choosing exactly one vertex per cluster.

Related work is to be found in [2] where Dror *et al.* present the generalized version of several combinatorial optimization problems including the generalized traveling salesman problem, the generalized Steiner tree problem, the generalized assignment problem, etc.

2. PROBLEM DESCRIPTION

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 $l \neq k$) and denote by $K = \{1, \ldots, m\}$ the index of the clusters. We assume that that edges are defined only between nodes which belong to different clusters and we denote the cost of an edge $e = (i, j) \in E$ by c_{ij} or by $c(i, j)$.

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 [7], Myung *et al.* proved that the GMST problem is $\mathcal{NP}$-hard. We proved in [9] a stronger result:

THEOREM 1. *The Generalized Minimum Spanning Tree problem on trees is $\mathcal{NP}$-hard.*

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

3. A STRONG INTEGER PROGRAMMING FORMULATION OF THE GMST PROBLEM

The GMST problem can be formulated as an integer program in many different ways, cf. [3], [4], [7], [8] and [9]. For example, in [8] we proposed a mixed integer programming formulation of the GMST problem, called the *bidirectional multicommodity flow formulation.* In that model a cluster $V_1 \subset V$ is chosen to be the *source* offering $|K| - 1 = m - 1$ commodities, one demanded by each of the remaining $m - 1$ clusters. Variables f_{ij}^k, $(i, j) \in A$ and $k = 1, ..., m - 1$ indicate the flow amount of commodity k going through the arc (i, j). Binary variables x_{ij}, with $(i, j) \in E$ and z_i, with $i \in V$ control the inclusion ($x_{ij} = 1$) or not ($x_{ij} = 0$) of edge (i, j), respectively the inclusion

($z_i = 1$) or not ($z_i = 0$) of node i in the solution. The GMST problem can be formulated as follows:

$$\min \sum_{(i,j)\in E} c_{ij}x_{ij}$$

(1) $\quad s.t.\quad f_{ij}^k + f_{ji}^{k'} \le x_e, \qquad \forall\, e=(i,j)\in E,\ \forall\, k,k'\in K_1$

(2) $\quad x(E) = m-1$

(3) $\quad z(V_k) = 1, \qquad \forall\, k\in K=\{1,...,m\}$

(4) $\quad \sum_{(i,j)\in A} f_{ij}^k \ge z_i, \qquad \forall\, k\in K_1,\ \forall\, i\in V_1$

(5) $\quad \sum_{(j,i)\in A} f_{ji}^k \ge z_i, \qquad \forall\, k\in K_1,\ \forall\, i\in V_k$

(6) $\quad \sum_{(i,j)\in A} f_{ij}^k - \sum_{(h,i)\in A} f_{hi}^k \ge 0,\ \forall\, k\in K_1,\ \forall\, i\in V\setminus(V_1\cup V_k)$

(7) $\quad f_{ij}^k \in \{0,1\}, \qquad \forall\, k\in K_1,\ \forall\,(i,j)\in A$

(8) $\quad x_{ij}, z_i \in \{0,1\}, \qquad \forall\,(i,j)\in E,\ \forall\, i\in V.$

Constraints (1) allow a non-zero flow f_{ij}^k or $f_{ji}^{k'}$ of commodity k or k' through an edge $e=(i,j)$ only if the latter is included in the solution. Constraints (2) and (3) guarantee that any feasible solution has $m-1$ edges and contains exactly one vertex from every cluster. Equations (4), (5) and (6), for a given $k\in K_1$, are the network flow equations for the problem of sending a flow of value 1 from cluster V_1 to cluster V_k. Note here that in this program we implicitly set $f_{ij}^k = 0$, $\forall\, k\in K_1$, $\forall\,(i,j)\in A$ with $j\in V_1$ (i.e. delete all the arcs in cluster V_1).

4. DEFINING A LAGRANGIAN PROBLEM

Lagrangian relaxation [1] is a well-known technique to find lower bounds for hard minimization problems in combinatorial optimization. The general idea is to "relax" (dualize) some (or all) constraints by adding them to the objective function using Lagrangian multipliers. Choosing values for the Lagrangian multipliers is of key importance in terms of quality of the lower bound generated.

We relax equations (4), (5) and (6) in a Lagrangian fashion. Let

$$t_{ik} \quad (\ge 0,\ \forall\, i\notin V_1\cup V_k,\ \forall\, k\in K_1)$$

be the Lagrange multipliers corresponding to (6),

$$t_{ik}^{(1)} \quad (\ge 0,\ \forall\, i\in V_1,\ \forall\, k\in K_1)$$

be the Lagrange multipliers corresponding to (4) and let

$$t_{ik}^{(k)} \quad (\geq 0,\ \forall\, i \in V_k,\ \forall\, k \in K_1)$$

be the Lagrange multipliers corresponding to (5).

Then the coefficient C_{ij}^k of f_{ij}^k in the objective function of the Lagrangian dual program is given by

$$C_{ij}^k \;=\; \begin{cases} -t_{ik}^{(1)} - t_{jk}^{(k)} & \text{if } i \in V_1 \text{ and } j \in V_k \\ -t_{ik} - t_{jk}^{(k)} & \text{if } i \notin V_1 \cup V_k \text{ and } j \in V_k \\ -t_{ik}^{(1)} + t_{jk} & \text{if } i \in V_1 \text{ and } j \notin V_1 \cup V_k \\ -t_{ik} + t_{jk} & \text{if } i \notin V_1 \cup V_k \text{ and } j \notin V_1 \cup V_k \\ 0 & \text{otherwise,} \end{cases}$$

the coefficient of z_i, p_i is given by

$$p_i \;=\; \begin{cases} \displaystyle\sum_{k=2}^{m} t_{ik}^{(1)} & \text{if } i \in V_1 \\ \displaystyle\sum_{k=2}^{m} t_{ik}^{(k)} & \text{if } i \in V_k, \end{cases}$$

and the Lagrangian dual program is:

$$\begin{aligned} \min \quad & \sum_{(i,j)\in E} c_{ij}x_{ij} + \sum_{(i,j)\in A}\sum_{k\in K_1} C_{ij}^k f_{ij}^k + \sum_{i\in V} p_i z_i \\ s.t. \quad & f_{ij}^k + f_{ji}^{k'} \leq x_e, \qquad \forall\, e=(i,j)\in E,\ \forall\, k,k' \in K_1 \\ & x(E) = m-1 \\ & z(V_k) = 1, \qquad \forall\, k \in K = \{1,...,m\} \\ & f_{ij}^k \in \{0,1\}, \qquad \forall\, k \in K_1,\ \forall\, (i,j) \in A \\ & x_{ij}, z_i \in \{0,1\}, \qquad \forall\, (i,j) \in E,\ \forall\, i \in V. \end{aligned}$$

From constraint (1), it is simple to deduce that the best contribution to the dual objective function is given by:

$$b_{ij} := c_{ij} + \sum_{k\in K_1} \min\{0, C_{ij}^k, C_{ji}^k\}$$

and therefore the Lagrangian dual program becomes:

$$\begin{aligned} \min \quad & \sum_{(i,j)\in E} b_{ij}x_{ij} + \sum_{i\in V} p_i z_i \\ & x(E) = m-1 \\ & z(V_k) = 1, \qquad \forall\, k \in K = \{1,...,m\} \\ & x_{ij}, z_i \in \{0,1\}, \qquad \forall\, (i,j) \in E,\ \forall\, i \in V. \end{aligned}$$

Let (X_{ij}), (Z_i), (F_{ij}^k), represent the optimum values of (x_{ij}), (z_i), (f_{ij}^k) in the solution of the Lagrangian dual program; then the optimal value of the Lagrangian dual program Z_D (a lower bound on the optimal solution of the GMST problem) is given by:

$$Z_D = \sum_{\{i,j\}\in E} b_{ij} X_{ij} + \sum_{i\in V} p_i Z_i.$$

5. THE SUBGRADIENT PROCEDURE

Choosing values for the Lagrangian multipliers is of key importance in terms of quality of lower bound generated by solving the Lagrangian dual problem.

In this section we use the subgradient method in an attempt to maximize the lower bounds obtained from the Lagrangian relaxation of the GMST problem. The procedure is as follows:

- **Step 1.** Set the initial values for the Lagrangian multipliers:

$$t_{ik}^{(1)} = 0,\ \forall\, i \in V_1,\ \forall\, k \in K_1,$$

$$t_{ik}^{(k)} = 0,\ \forall\, i \in V_k,\ \forall\, k \in K_1,$$

$$t_{ik} = 0,\ \forall\, i \notin V_1 \cup V_k,\ \forall\, k \in K_1,$$

  and initialize Z_{UB}, the upper bound of the problem (e.g. from some heuristic for the problem).
- **Step 2.** Solve the Lagrangian dual program with the current set of multipliers and let the solution be Z_D, (X_{ij}), (Z_i), (F_{ij}^k).
- **Step 3.** If the Lagrangian solution (X_{ij}), (Z_i), (F_{ij}^k) is a feasible solution to the original problem then update Z_{UB}, the upper bound on the problem corresponding to a feasible solution, accordingly. Update Z_{max} at each subgradient iteration using $Z_{max} = \max\{Z_{max}, Z_D\}$, where Z_{max} denotes the maximum lower bound found over all subgradient iterations. Initially $Z_{max} = -\infty$.
- **Step 4.** Stop if $Z_{UB} = Z_{max}$ since then Z_{UB} is the optimal solution, else go to Step 5.
- **Step 5.** Calculate the subgradients

$$H_{ik}^{(1)} = Z_i - \sum_{(i,j)\in A} F_{ij}^k, \qquad \forall\, i \in V_1,\ \forall\, k \in K_1$$

$$H_{ik}^{(k)} = Z_i - \sum_{(j,i)\in A} F_{ji}^k, \qquad \forall\, i \in V_k,\ \forall\, k \in K_1$$

$$H_{ik} = \sum_{(j,i)\in A} F_{ji}^k - \sum_{(i,j)\in A} F_{ij}^k, \forall\, i \notin V_1 \cup V_k,\ \forall\, k \in K_1.$$

- **Step 6.** Define a step size T by
$$T = \alpha \frac{(Z_{UB} - Z_D)}{\|H\|},$$
where $0 < \alpha \leq 2$, and $\|H\|$ is defined by
$$\|H\| = \sum_{i \in V_1} \sum_{k \in K_1} (H_{ik}^{(1)})^2 + \sum_{i \in V_k} \sum_{k \in K_1} (H_{ik}^{(k)})^2 + \sum_{i \notin V_1 \cup V_k} \sum_{k \in K_1} (H_{ik})^2.$$
This step size depends upon the gap between the current lower bound Z_D and the upper bound Z_{UB} and the user defined parameter α with $\|H\|$ being a scaling factor.
- **Step 7.** Update the Lagrange multipliers by
$$t_{ik}^{(1)} = \max\{0, t_{ik}^{(1)} + TH_{ik}^{(1)}\}, \ \forall\, i \in V_1, \ \forall\, k \in K_1,$$
$$t_{ik}^{(k)} = \max\{0, t_{ik}^{(k)} + TH_{ik}^{(k)}\}, \ \forall\, i \in V_k, \ \forall\, k \in K_1,$$
$$t_{ik} = \max\{0, t_{ik} + TH_{ik}\}, \ \forall\, i \notin V_1 \cup V_k, \ \forall\, k \in K_1,$$
- **Step 8.** Go to Step 2 to resolve the Lagrangian dual program with this new set of multipliers unless a stopping criterium is met.

Initially we set $\alpha = 2$. If Z_{max} is not improved (i.e. increased) in the last N subgradient iterations with the current value of α then we halve α. Based on experimenting (computationally) with different values of N, it seems to be reasonable to choose the value $N = 30$.

In our experiments we set $Z_{UB} = Z_{OPT}$, where Z_{OPT} is the optimal value of the GMST problem calculated using the rooting procedure (see [8]).

6. COMPUTATIONAL RESULTS

According to the method of generating the edge costs, the problems generated are classified into three types:

- **structured Euclidean case**
- **unstructured Euclidean case**
- **non-Euclidean case**.

For the instances in the structured Euclidean case m squares (clusters) are "packed in a square" and in each of these m clusters n_c nodes are selected randomly. The costs between nodes are the Euclidean distances between the nodes. So in this model the clusters can be interpreted as physical clusters. In the other models such an interpretation is not valid.

For the unstructured Euclidean case $n = mn_c$ nodes are generated randomly in $[0, 100]^2$ with costs given by the Euclidean distances. But then the clusters are choosen randomly among these points. Finally in the non-Euclidean model the edge costs are randomly generated on $[0, 100]$.

Our computational experiments were performed on a HP 9000/735 computer with a 125 MHz processor and 144 MB memory. Our Lagrangian relaxation scheme is written in C and compiled with a HP-UX cc compiler.

Table 1: Computational Results for unstructured Euclidean problems

Problem number	m	n_c	OPT	LB	Gap [%]	CPU [s]	no. subgradient iterations
1	8	3	90	89.998	0.00	0.09	25
2		4	88	87.996	0.00	0.12	30
3		6	60	59.999	0.00	0.43	65
4		8	46	45.941	0.13	36.93	160
5		10	45	44.953	0.10	48.18	180
6	10	3	108	107.998	0	7.08	35
7		4	91	90.976	0.03	16.14	50
8		6	85	84.972	0.03	20.73	65
9		8	67	66.951	0.07	41.83	125
10		10	62	61.943	0.09	55.27	140
11	12	3	116	115.947	0.05	13.03	45
12		4	107	106.916	0.08	24.94	75
13		6	89	88.836	0.18	78.53	150
14		8	75	74.759	0.32	174.43	275
15	15	3	135	134.987	0.01	13.72	30
16		4	124	123.985	0.01	16.91	45
17		6	96	95.899	0.11	72.88	105
18		8	91	90.869	0.14	134.29	175
19	18	3	153	152.991	0.01	15.29	40
20		4	132	131.985	0.01	25.65	45
21		6	127	126.909	0.07	84.28	110
22		8	112	111.849	0.14	204.12	215
23	20	3	175	174.981	0.01	16.39	35
24		4	147	146.972	0.02	44.71	75
25		6	129	128.959	0.04	65.78	95
26		8	108	107.795	0.19	220.43	200
27	25	3	182	181.975	0.01	23.79	60
28		4	159	158.969	0.02	61.55	90
29		6	142	141.893	0.08	92.37	125
30		8	127	126.825	0.14	289.78	250
31	30	3	196	195.926	0.04	41.27	75
32		4	174	173.889	0.06	101.35	150

Table 2: Computational Results for structured Euclidean problems

Problem number	m	n_c	OPT	LB	Gap [%]	CPU [s]	no. subgrad. iterations
1	8	3	983	982.998	0.00	0.08	25
2		4	966	965.993	0.00	0.18	30
3		6	960	959.988	0.00	0.49	65
4		8	934	933.953	0.01	36.91	160
5		10	922	921.949	0.01	48.72	180
6	10	3	1251	1250.998	0.00	7.23	35
7		4	1243	1242.980	0.00	16.59	50
8		6	1240	1239.975	0.00	20.93	65
9		8	1225	1224.956	0.00	41.98	125
10		10	1208	61.943	0.00	55.31	140
11	12	3	1616	1615.951	0.00	14.63	45
12		4	1545	1544.908	0.01	25.76	75
13		6	1487	1486.841	0.01	78.96	150
14		8	1458	1457.745	0.02	174.81	275
15	15	3	1977	1976.985	0.00	14.79	30
16		4	1966	1965.981	0.00	18.85	45
17		6	1946	1945.909	0.00	75.81	105
18		8	1932	1931.899	0.01	138.77	175
19	18	3	2384	2383.992	0.00	17.16	40
20		4	2365	2364.985	0.00	27.70	45
21		6	2352	2351.904	0.00	89.22	110
22		8	2338	2337.863	0.01	225.08	215
23	20	3	2765	2764.972	0.00	17.32	35
24		4	2741	2740.968	0.00	46.81	75
25		6	2728	2727.956	0.00	69.77	95
26		8	2709	2708.797	0.01	231.48	200
27	25	3	3112	3111.967	0.00	24.72	60
28		4	3094	3093.958	0.00	64.51	90
29		6	3075	3074.795	0.01	97.38	125
30		8	3058	3057.715	0.01	298.89	250
31	30	3	3478	3477.832	0.00	52.13	75
32		4	3452	3451.787	0.01	138.96	150

Table 3: Computational Results for non-Euclidean problems

Problem number	m	n_c	OPT	LB	Gap [%]	CPU [s]	no. subgradient iterations
1	8	3	28	27.998	0.01	0.05	25
2		4	19	18.996	0.02	0.11	25
3		6	11	10.995	0.05	0.42	65
4		8	13	12.952	0.37	35.02	150
5		10	9	8.953	0.52	40.12	175
6	10	3	29	28.998	0.01	7.03	35
7		4	24	23.986	0.06	15.01	50
8		6	16	15.985	0.09	20.17	65
9		8	12	11.956	0.37	40.73	110
10		10	11	10.941	0.54	43.28	120
11	12	3	34	33.999	0.00	12.25	45
12		4	21	20.998	0.01	24.73	75
13		6	20	19.866	0.67	76.28	150
14		8	14	13.759	1.75	172.32	250
15		10	12	11.825	1.48	177.23	275
16	15	3	28	27.996	0.01	12.89	30
17		4	25	24.993	0.01	15.23	45
18		6	19	18.967	0.17	70.12	105
19		8	15	14.823	1.19	135.95	180
20		10	14	13.792	1.46	142.37	210
21	18	3	34	33.994	0.02	14.61	45
22		4	34	33.995	0.01	21.75	40
23		6	22	21.974	0.07	83.27	110
24		8	18	17.921	0.14	165.38	200
25		10	17	16.823	0.14	171.82	215
26	20	3	37	36.989	0.03	16.78	35
27		4	28	27.972	0.10	43.66	75
28		6	27	26.982	0.07	59.23	85
29		8	19	18.899	0.53	201.83	175
30		10	19	18.792	1.11	200.25	180
31	25	3	46	45.978	0.05	51.29	75
32		4	35	34.981	0.05	60.07	80
33		6	24	23.829	0.72	169.23	180
34		8	24	23.779	0.93	202.78	230
35	30	3	43	42.969	0.07	53.21	75
36		4	32	31.978	0.07	63.72	80
37		6	29	23.852	0.62	174.55	170
38		8	29	28.872	0.44	209.33	220
39	40	3	44	43.967	0.08	82.51	80
40		4	41	40.915	0.21	101.73	90

Each line corresponds to an instance. The first column is the problem number. The next two columns give the number of clusters m and the number of nodes per cluster n_c. The fourth column (OPT) contains the optimal value of the GMST problem found by using the rooting procedure (see [8]). The fifth column (LB) gives the lower bound obtained using the subgradient method. The next column (GAP %) gives the gap in percentage defined by $100(OPT - LB)/LB$. The last two columns give the CPU time and the number of subgradient iterations necessary to find the lower bound.

Comparing these results with the lower bounds provided by the LP relaxation of the GMST problem (we did numerical experiments which are not given here, solving the LP relaxation of the GMST problem by CPLEX), the lower bounds obtained using our Lagrangian relaxation scheme are in general better.

Remark 1. Here our experiments were made by setting $Z_{UP} = Z_{OPT}$, where Z_{OPT} is the optimal value of the GMST problem calculated using the rooting procedure (see Section 3.6). In practice the optimal value Z_{OPT} is unknown, therefore we only may have an upper bound $Z_{UP} \approx Z_{OPT}$. But numerical experiments have shown that even in this case the quality of the gap is maintained. The values of the gap in percentage in the special non-Euclidean case $m = 15$ and $n_c = 6$, using different upper bounds, are reported in the next table.

m	n_c	OPT	UB	Gap [%)]	CPU [s]	no. subgrad. iterations
15	6	19	–	0.17	70.12	105
15	6	–	19.5	0.19	70.33	105
15	6	–	20	0.20	70.67	105
15	6	–	21	0.23	71.09	105

The first two columns give the number of clusters m and the number of nodes per cluster n_c. The third column (OPT) contains the optimal value of the GMST problem found by using the rooting procedure (see Section 3.6). The forth column (UB) gives the upper bound. The next column (GAP %) gives the gap in percentage defined by $100(UB - LB)/LB$. The last two columns give the CPU time and the number of subgradient iterations.

7. CONCLUDING REMARKS

We presented a Lagrangian relaxation of a bidirectional multicommodity flow formulation of the GMST problem. The subgradient optimization algorithm was used to obtain lower bounds.

Computational experiments (by setting $Z_{UP} = Z_{OPT}$) show that using our Lagrangian relaxation scheme are in general better than the lower bounds provided by the linear programming relaxation of the GMST problem. This result holds even in the case when we may have available an upper bound $Z_{UP} \approx Z_{OPT}$.

REFERENCES

[1] Bertsimas, D. and Tsitsiklis, J.N., *Introduction to Linear Optimization*, Athena Scientific, Belmont, Massachusetts, 1997.

[2] Dror, M., Haouari, M. and Chaouachi, J., *Generalized Spanning Trees*, European Journal of Operational Research, **120** (2000), 583–592.

[3] Feremans, C., *Generalized Spanning Trees and Extensions*, PhD thesis, Universite Libre de Bruxelles, Belgium, 2002.

[4] Feremans, C, Labbe, M. and Laporte, G., *A Comparative Analysis of Several Formulations of the Generalized Minimum Spanning Tree Problem*, Networks Vol. **39** (2002), 29–34.

[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] Gerla M. and Frata, L., *Tree structured fiber optics MAN's*, IEEE J. Select. Areas Comm. SAC, **6** (1988), 934–943.

[7] Myung, Y.S., Lee, C.H. and Tcha, D.W., *On the Generalized Minimum Spanning Tree Problem*, Networks, **26** (1995), 231–241.

[8] Pop, P.C., *The Generalized Minimum Spanning Tree Problem*, PhD thesis, University of Twente, The Netherlands, 2002.

[9] Pop, P.C., *New Models of the Generalized Minimum Spanning Tree Problem*, Journal of Mathematical Modelling and Algorithms, **3** (2004), 153–166.

[10] Pop, P.C., *A New Relaxation Method for the Generalized Minimum Spanning Tree Problem*, to appear in European Journal of Operations Research.

[11] Prisco, J.J., *Fiber optic regional area networks in New York and Dallas*, IEEE J. Select. Areas Comm. SAC, **4** (1986), 750–757.

Received March 21, 2005

Faculty of Sciences,
North University of Baia Mare
Department of Mathematics and Computer Science
Baia Mare, 4800, Romania
E-mail: pop_petrica@yahoo.com