Paper

Extensions of the minimum labelling spanning tree problem

Raffaele Cerulli, Andreas Fink, Monica Gentili, and Stefan Voß

Abstract— In this paper we propose some extensions of the minimum labelling spanning tree problem. The main focus is on the minimum labelling Steiner tree problem: given a graph *G* with a color (label) assigned to each edge, and a subset *Q* of the nodes of *G* (basic vertices), we look for a connected subgraph of *G* with the minimum number of different colors covering all the basic vertices. The problem has several applications in telecommunication networks, electric networks, multimodal transportation networks, among others, where one aims to ensure connectivity by means of homogeneous connections. Numerical results for several metaheuristics to solve the problem are presented.

Keywords— network design, metaheuristics, spanning trees, labelling trees, Steiner tree problem.

1. Introduction

Many real-world problems can be modelled by means of graphs where a label or a weight is assigned to each edge and the aim is to optimize a certain function of these weights. In particular, one can think of problems where the objective is to find homogeneous subgraphs (respecting certain connectivity constraints) of the original graph. This is the case, e.g., for telecommunication networks (and, more generally, any type of communication networks) that are managed by different and competing companies. The aim of each company is to ensure the service to each terminal node of the network by minimizing the cost (i.e., by minimizing the use of connections managed by other companies).

This kind of problem can be modelled as follows. The telecommunication network is represented by a graph $G = (V,E)$ where with each edge $e \in E$ is assigned a set of colors L_e and each color denotes a different company that manages the edge. The aim of each company is to define a spanning tree of G that uses the minimum number of colors. When the graph represents a transportation network and the colors, assigned to each edge, represent different modes of transportation, then looking for a path that uses the minimum number of colors from a given source s to a given destination t means to look for a path connecting s and t using the minimum number of different modes of transportation.

We focus on the *minimum labelling Steiner tree problem* (MLSteiner): given a graph $G = (V,E)$, with a label (color) assigned to each edge and a subset $Q \subseteq V$ of nodes of G (basic vertices or nodes), we look for an acyclic connected subgraph of G spanning all basic nodes and using the minimum number of different colors. This problem is an extension of the *minimum labelling spanning tree problem* (MLST): given a graph G with a label (color) assigned to each edge we look for a spanning tree of G with the minimum number of different colors.

In this paper, first we review the earlier results existing in the literature to solve the MLST. Then we discuss how these approaches can be easily extended to efficiently solve the MLSteiner and present a comprehensive study of experimental results.

The sequel of the paper is organized as follows. Section 2 summarizes existing approaches for the MLST as well as some important references on the Steiner problem in graphs. In Section 3 we sketch some extensions of the MLST related to the MLSteiner which is the focus of this study. Section 4 presents our modifications of the solution approaches for the MLST to solve the MLSteiner. In Section 5 we present experimental results, and, finally, Section 6 gives some further research options.

2. Literature review

2.1. Earlier approaches to solve the MLST

The MLST was initially addressed by Broersma and Li [2]. They proved, on the one hand, that the MLST is $\mathcal{NP}$-hard by reduction from the minimum dominating set problem, and, on the other hand, that the "opposite" problem of looking for a spanning tree with the maximum number of colors is polynomially solvable. Independently, Chang and Leu [6] provided a different $\mathcal{NP}$-hardness proof of the problem by reduction from the set covering problem. They also developed two heuristics to determine feasible solutions of the problem and tested the performance of these heuristics by comparison with the results of an exact approach based on an A^* algorithm.

Krumke and Wirth [14] formulated an approximation algorithm (in the sequel referred to as maximum vertex covering algorithm – MVCA) with logarithmic performance guarantee and showed also that the problem cannot be approximated within a constant factor. Wan *et al*. [19] provided a better analysis of the greedy algorithm given in [14] by showing that its worst case performance ratio is at most $\ln(n-1)+1$ where n denotes the number of nodes of the given graph, i.e., $n = |V|$. Recently, Xiong *et al*. [23] obtained the better bound $1+\ln b$ where each color appears at most b times. Moreover, Xiong *et al*. [21] proposed a genetic algorithm to solve the MLST and provided some experimental results.

In [5] we presented several metaheuristic approaches to solve the MLST (namely, simulated annealing, reactive tabu search, the pilot method and variable neighborhood search) and compared them with the results provided by the MVCA heuristic presented in [14, 21]. Recently, a modification of our pilot method combined with the genetic algorithm of Xiong *et al*. [21] was shown to be effective by [22].

A variant of the problem has been studied by Brüggemann *et al*. [3], where the MLST with bounded color classes has been addressed. In this variant, each color of the graph is assumed to appear at most r times. This special case of the MLST is polynomially solvable for $r = 2$, and $\mathcal{NP}$-hard and APX-complete for $r \geq 3$. Local search algorithms for this variant, that are allowed to switch up to k of the colors used in a feasible solution have been studied, too. For $k = 2$, the authors showed that any local optimum yields an $\frac{(r+1)}{2}$-approximation of the global optimum, and this bound is best possible. For every $k \geq 3$, there exist instances for which some local optimum is a factor of $\frac{r}{2}$ away from the global optimum.

2.2. The Steiner tree problem

Consider an undirected connected graph $G = (V,E)$ with node set V, edge set E, and nonnegative weights associated with the edges. Given a set $Q \subseteq V$ of specified vertices (called terminals or basic vertices) *Steiner's problem* in graphs (SP) is to find a minimum cost subgraph of G such that there exists a path in the subgraph between every pair of basic vertices. In order to achieve this minimum cost subgraph additional vertices from the set $S := V \setminus Q$, called Steiner vertices, may be included. Since all edge weights are assumed to be nonnegative, there is an optimal solution which is a tree, called Steiner tree.

Correspondingly, *Steiner's problem in directed* graphs (SPD) is to find a minimum cost directed subgraph of a given graph that contains a directed path between a root node and every basic vertex. Applications of the SP and the SPD are frequently found in many problems related to network design and telecommunications. Beyond that, SP and SPD have equal importance also for the layout of connection structures in networks as, e.g., in topological network design, location science and VLSI (very large scale integrated circuits) design.

The SP is a well-studied problem and there is a wealth of excellent reference providing information on Steiner problems, such as [11]. Additional surveys on quite broad aspects of Steiner tree problems are provided by [10, 16, 20] as well as, most recently, [17].

3. Extension of the MLST

Steiner tree problems refer to important problem classes in graphs. The SP may be called one of the most important combinatorial optimization problems. Modifications and generalizations of Steiner tree problems will certainly arise and become a core focus of research and telecommunications applications including additional online optimization problems as well as stochastic optimization approaches. In that sense we have defined the MLSteiner as an extension of both, the MLST as well as the SP. But in this section we go beyond this.

Examples for possible generalizations may include, e.g., a weighted labelling Steiner tree problem with budget constraints. Here we are given a graph G with a label (color) assigned to each edge and we look for a spanning tree with respect to a given subset Q of the nodes of G with the minimum number of different colors. Furthermore, one may incorporate some weights on the edges and define a budget constraint on the sum of the weights of included edges while still minimizing the number of labels or more versatile capacitated Steiner tree problems.

Other ideas on generalizations of the MLST refer to certain ring network design problems with or without budget constraints (see, e.g., [8]) that may be formulated in terms of minimizing the number of labels once they are to be defined and considered. While these generalizations may prove to be important, subsequently we focus on the MLSteiner.

4. Different metaheuristic approaches to solve the MLSteiner

We aim for a Steiner tree which connects all required or basic nodes with a minimum number of colors/labels. Within the steps of the search process only those edges that are colored according to the currently activated colors may be used. For describing different metaheuristic approaches for the MLSteiner we heavily rely on our previous approaches for the MLST described in [5]. We have adapted our code for the MLST so that the algorithm checks whether the resulting subgraph (restricted to the edges with actually used or activated colors) connects all required nodes. If there are disconnections, large penalty values are added to the objective function so that the search process is directed towards feasibility.

Before going into detail let us introduce some notation. Given an undirected graph $G = (V,Q,E)$ with V being the set of nodes, $Q \subseteq V$ the subset of basic nodes and E denoting the set of edges, let c_e be the color (label) associated with edge $e \in E$ and $L = \{c_1, c_2, \ldots, c_l\}$ be the set of all colors. We denote by $C(F) = \bigcup_{e \in F} c_e$ the set of colors assigned with edges in $F \subseteq E$. Any subgraph T of G can be represented by the set of its colors $C(T)$. Given a set of colors C, we define by $V(C)$ the subset of nodes of G covered by the edge set defined by C, i.e., $V(C) = \{i \in V : e \in E$ is incident to i and $c_e \in C\}$. A set of colors C is *feasible* for the MLSteiner if and only if the corresponding set of edges defines a connected subgraph $G_C = (V', E')$ that spans all the basic nodes of G, i.e., $V' \cap Q = Q$. (Moreover, we note in passing that we assume $|L_e| = 1$ throughout the remainder of this paper. That is, exactly one color is assigned to each edge.).

4.1. Greedy

The algorithm starts with an empty set of edges. Then, it iteratively selects one color among the unused ones and inserts all edges of that color in the graph until all the basic nodes are connected. At each iteration it tests all the unused colors and chooses a color in that way that the decrease in the number of Steiner connected components is as large as possible, where we define a Steiner connected component as a connected component $H = (V', E')$ of the graph that contains at least one basic node, e.g., $V' \cap Q \neq \emptyset$. The proposed algorithm is illustrated below.

Algorithm: The greedy heuristic

Let $C = \emptyset$ be the set of used colors.

Repeat
  let H be the subgraph of G restricted to edges with colors from C;
  let H' be the subgraph of H restricted to the Steiner connected components of H;
  for all $c_i \in L \setminus C$ **do**
    determine the number of Steiner connected components when inserting all edges with color c_i in H;
  end for
  choose color c_i with the smallest resulting number of Steiner connected components and do: $C = C \cup \{c_i\}$;
until H' is connected.

The greedy strategy we adopt differs from the MVCA heuristic since it carries out operations on the Steiner connected components of subgraph H, while MVCA considers all the connected components of such a graph.

The running time of the proposed greedy strategy is $O(l^2 n)$, where l is the total number of different colors in G. Indeed, the *repeat* loop will take $O(l)$ steps and we have $O(ln)$ to carry out the *for*-loop.

Since the MLST is a special case of the MLSteiner, then, by applying the same reasoning introduced in [19], we can derive the following approximation result.

Theorem 1: Given any MLSteiner instance with n nodes and q basic nodes ($q < n$, $n > 1$), the greedy algorithm provides an $(\ln(q-1)+1)$-approximation.

4.2. Variable neighborhood search

Variable neighborhood search (VNS) goes back to Mladenović and Hansen [15]. The underlying idea of VNS is to generalize the classical local search based approaches by considering a multi-neighborhood structure, i.e., a set of pre-selected neighborhood structures $\mathcal{N} = \{N_1, N_2, \ldots, N_s\}$ such that $N_j(C)$, $j = 1, 2, \ldots, s$ is the set of solutions in the jth neighborhood of C. The basic VNS algorithm, applied to solve the MLSteiner, is described below.

Algorithm: The basic VNS algorithm

Step 1. Consider an initial feasible solution $C \subseteq L$ and set $k \leftarrow 1$.

Step 2. Generate at random a solution $C' \in N_k(C)$.

Step 3. Apply a local search algorithm, starting from the initial solution C', to obtain a local optimum C''.

Step 4. If $|C''| < |C|$ then: $C \leftarrow C''$ and set $k \leftarrow 1$ otherwise $k \leftarrow k+1$.

Step 5. If $k \leq k_{\max}$ then go to Step 1, else Stop.

We implemented VNS by using three different neighborhood structures, in order to check whether one neighborhood is better than the other. In particular, given a feasible color set C, we consider the following neighborhood structures:

- ***k – switch neighborhood*** $N_k^1(C)$
  A set $C' \in N_k^1(C)$ if and only if we can get the color set C' from the color set C by removing up to k colors from C and adding up to k new colors. That is, $N_k^1(C) = \{C' \subseteq L : |C' \backslash C| \leq k \text{ and } |C \backslash C'| \leq k\}$.

- ***k – covering neighborhood*** $N_k^2(C)$
  A set $C' \in N_k^2(C)$ if and only if the common colors between C and C' cover at least k basic nodes. That is, $N_k^2(C) = \{C' \subseteq L : |V(C' \cap C) \cap Q| \geq k$ and $|C'| \leq |C|\}$.

- ***k – mixed neighborhood*** $N_k^3(C)$
  A set $C' \in N_k^3(C)$ if and only if C' contains exactly $|C| - k$ colors in common with C and all the remaining different colors cover a greater number of basic vertices. That is, $N_k^3(C) = \{C' \subseteq L : |C \backslash C'| = k$, $|C' \backslash C| \leq k$ and $|V(C \backslash C') \cap Q| \leq |V(C' \backslash C) \cap Q|\}$.

For each of the neighborhood structures described above, the procedure starts from an initial feasible solution C provided by the greedy algorithm described in Subsection 4.1. At each generic iteration the VNS:

– selects at random a feasible solution C' in the neighborhood $N_k^i(C)$;

– applies a local exchange strategy that, for a maximum number $h_{\max}$ of iterations, tries to decrease the size of C' to obtain a possible better solution C'' by removing π labels and adding up to π new labels, where $\pi = 2, 3, \ldots, |C'|$;

– defines the new neighborhood to be explored in the next iteration.

In our implementation of VNS, we let parameter $k_{\max}$ vary during the execution, that is $k_{\max} = \min\{|C|, \frac{l}{4}\}$, where $|C|$ is the "size" of the current feasible solution whose neighborhood is being explored.
In the sequel we refer to the implementations of VNS using $N_k^1()$, $N_k^2()$ and $N_k^3()$ as VNS1, VNS2 and VNS3, respectively.

4.3. Simulated annealing

Simulated annealing (SA) extends basic local search by allowing moves to worse solutions [13]. The basic concept of SA is the following: starting from an initial solution (in our implementation from an empty set of activated colors as in the greedy heuristic), successively, a candidate move is randomly selected. This move is accepted if it leads to a solution with a better objective function value than the current solution, otherwise the move is accepted with a probability that depends on the deterioration Δ of the objective function value. The acceptance probability is computed according to the Boltzmann function as $e^{-\Delta/T}$, using a temperature T as control parameter.
Following [12], the value of T is initially high, which allows many worse moves to be accepted, and is gradually reduced through multiplication by a parameter *cooling factor* according to a geometric cooling schedule. Given a parameter *size factor*, *size factor* $\times\, l$ candidate moves are tested (note that l denotes the neighborhood size) before the temperature is reduced. The starting temperature is determined as follows: given a parameter *initial acceptance fraction* and based on an abbreviated trial run, the starting temperature is set so that the fraction of accepted moves is approximately *initial acceptance fraction*. A further parameter, *frozen acceptance fraction* is used to decide whether the annealing process is *frozen* and should be terminated. Every time a temperature is completed with less than *frozen acceptance fraction* of the candidate moves accepted, a counter is increased by one, while this counter is re-set to 0 each time a new best solution has been obtained. The whole procedure is terminated when this counter reaches a parameter *frozen limit*. For our implementation we follow the parameter setting of [12], which was reported to be robust for various problems. That is, we use $\alpha = 0.95$, *initial acceptance fraction* = 0.4, *frozen acceptance fraction* = 0.02, *size factor* = 16 and *frozen limit* = 5.

4.4. Reactive tabu search

The basic paradigm of *tabu search* (TS) is to use information (in the sense of an adaptive memory) about the search history to guide local search approaches to overcome local optimality (see [9] for a survey on tabu search). In general, this is done by a dynamic transformation of the local neighborhood. Based on some sort of memory certain moves may be forbidden, they are defined tabu (and appropriate move attributes such as a certain index indicating a specific color put into a list, called tabu list). As for SA, the search may imply acceptance of deteriorating moves when no improving moves exist or all improving moves of the current neighborhood are set tabu. At each iteration a best admissible neighbor may be selected. A neighbor, respectively a corresponding move, is called admissible, if it is not tabu.
Reactive TS (RTS) aims at the automatic adaptation of the tabu list length [1]. The idea is to increase the tabu list length when the tabu memory indicates that the search is revisiting formerly traversed solutions. A possible specification is the following. Starting with a tabu list length s of 1, it is increased to $\min\{\max\{s+2, s\times 1.2\}, b_u\}$ every time a solution has been repeated, taking into account an appropriate upper bound b_u (to guarantee at least one admissible move). If there is no repetition for some iterations, we decrease it to $\max\{\min\{s-2, s/1.2\}, 1\}$. To accomplish the detection of a repetition of a solution, one may apply a trajectory based memory using hash codes.
For RTS, it is appropriate to include means for diversifying moves whenever the tabu memory indicates that one is trapped in a certain basin of attraction. As a trigger mechanism one may use, e.g., the combination of at least three solutions each having been traversed three times. A very simple escape strategy is to perform randomly a number of moves (depending on the average of the number of iterations between solution repetitions). For our implementation of RTS we consider as initial solution (as for the SA and the greedy heuristic) an empty set of activated colors. As termination criterion we consider a given time limit.

4.5. Pilot method

Using a greedy construction heuristic such as the MVCA as a building block or application process, the pilot method is a metaheuristic with the primary idea of performing repetition exploiting the application process as a look ahead mechanism [7, 18]. In each iteration (of the pilot method) one tentatively determines for every possible local choice (i.e., move to a neighbor of the current solution, called master solution) a look ahead or pilot solution, recording the best results in order to extend at the end of the iteration the master solution with the corresponding move. This strategy may be applied by successively performing, e.g., a construction heuristic for all possible local choices (i.e., starting a new solution from each incomplete solution that can result from the inclusion of any not yet included element into the current incomplete solution).
We apply the pilot method in connection with a greedy local search strategy operating on a solution space that includes incomplete (infeasible) solutions and a neighborhood that considers the addition of colors (see MVCA). We take into account infeasibilities by adding appropriate penalty values. The pilot method successively chooses the best local move (regarding the additional activation of one color) by evaluating such neighbors with a steepest descent until a local optimum, and with that a feasible solution, is obtained. (Note that as for the MVCA, at the end it may be beneficial to greedily drop colors while retaining feasibility.)

5. Experimental results

In this section we report some of our computational results. We considered different groups of instances in order to evaluate how the performance of the algorithms is influenced by both

– the number and distribution of the basic nodes;

– the distribution of the labels on the edges.

In particular, we defined different scenarios based on different parameter settings: n – number of nodes of the graph; l – total number of colors assigned to the graph; m – total number of edges of the graph computed by $m = \frac{d(n-1)n}{2}$, where d is a measure of density of the graph, and, q – the number of basic nodes of the graph. Parameter settings are: $n = 50, 100$, $l = 0.25n, 0.5n, n, 1.25n$, $d = 0.2, 0.5, 0.8$ and $q = 0.2n, 0.4n$, for a total of 48 different scenarios. For each scenario we generated ten different instances. All the generated data are available upon request from the authors.

Our results are reported in Tables 1–4. In each table the first three columns show the parameters characterizing the different scenarios (n, l, d, while the values of q determine the different tables). The remaining columns give the results of the greedy heuristic and of our metaheuristics: variable neighborhood search (VNS1, VNS2 and VNS3), simulated annealing, reactive tabu search and the pilot method, respectively. For the results reported on the greedy heuristic we note that we have implemented the idea to try at the end to greedily drop colors while retaining feasibility. In two of all 480 cases this reduced the objective value by 1.

In general we can say that the pilot method behaves best with respect to solution quality. The SA is usually outperformed by all other metaheuristics and RTS overall behaves a bit better than the VNS for VNS2 and VNS3. The RTS and VNS1 are somewhat incomparable as there does not seem to be a clear picture which method behaves best with respect to solution quality. Among the VNS implementations the first version seems to provide better results than the other two implementations.

Closer inspection of the results reveals a few probably unusual behaviours. First of all, we encountered a considerable role of how ties are broken. Assuming that a tie is broken unfavorably and one encounters an increase or decrease of the objective function by 1, the percentage deviation is affected considerably as most problem instances tend to have very small objective function values (considering the pilot method, only in 6 of the cases with $q = 0.2n$

Table 1
Computational results for $n = 50$ and $q = 0.2n$

n	l	d	Greedy	VNS1	VNS2	VNS3	SA	RTS	Pilot
50	12.5	0.8	1.0	1.0	1.0	1.0	1.0	1.0	1.0
50	12.5	0.5	1.1	1.1	1.1	1.1	1.1	1.1	1.1
50	12.5	0.2	2.1	2.0	2.0	2.1	2.0	2.0	2.0
50	25	0.8	1.2	1.2	1.2	1.2	1.2	1.2	1.2
50	25	0.5	1.9	1.9	1.9	1.9	1.9	1.9	1.9
50	25	0.2	2.9	2.9	2.9	2.9	2.9	2.9	2.9
50	50	0.8	2.0	2.0	2.0	2.0	2.0	2.0	2.0
50	50	0.5	2.8	2.8	2.6	2.7	2.7	2.8	2.6
50	50	0.2	4.0	3.9	3.9	4.0	4.0	4.0	3.9
50	62.5	0.8	2.3	2.0	2.0	2.0	2.2	2.2	2.0
50	62.5	0.5	3.1	2.8	2.9	3.0	3.0	3.0	2.8
50	62.5	0.2	4.4	4.3	4.4	4.5	4.4	4.4	4.3
Sum			28.8	27.9	27.9	28.4	28.4	28.5	27.7

Table 2
Computational results for $n = 100$ and $q = 0.2n$

n	l	d	Greedy	VNS1	VNS2	VNS3	SA	RTS	Pilot
100	25	0.8	1.0	1.0	1.0	1.0	1.0	1.0	1.0
100	25	0.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5
100	25	0.2	2.2	2.1	2.1	2.1	2.1	2.1	2.1
100	50	0.8	1.9	1.9	1.9	1.9	1.9	1.9	1.9
100	50	0.5	2.0	2.0	2.0	2.0	2.0	2.0	2.0
100	50	0.2	3.3	3.2	3.2	3.3	3.5	3.3	3.2
100	100	0.8	2.4	2.1	2.2	2.2	2.5	2.4	2.0
100	100	0.5	3.0	3.0	3.0	3.1	3.4	3.0	3.0
100	100	0.2	4.9	4.6	5.1	5.0	5.2	4.8	4.6
100	125	0.8	2.8	2.8	2.8	2.8	3.0	2.8	2.8
100	125	0.5	3.5	3.4	3.5	3.6	4.0	3.5	3.4
100	125	0.2	5.7	5.3	5.9	5.7	6.2	5.6	5.4
Sum			34.2	32.9	34.2	34.2	36.3	33.9	32.9

Table 3
Computational results for $n = 50$ and $q = 0.4n$

n	l	d	Greedy	VNS1	VNS2	VNS3	SA	RTS	Pilot
50	12.5	0.8	1.0	1.0	1.0	1.0	1.0	1.0	1.0
50	12.5	0.5	1.4	1.4	1.4	1.4	1.4	1.4	1.4
50	12.5	0.2	2.6	2.6	2.6	2.6	2.6	2.6	2.6
50	25	0.8	2.0	2.0	2.0	2.0	2.0	2.0	2.0
50	25	0.5	2.0	2.0	2.0	2.0	2.0	2.0	2.0
50	25	0.2	4.3	3.9	3.9	3.9	3.9	4.0	3.9
50	50	0.8	2.4	2.2	2.3	2.2	2.5	2.4	2.1
50	50	0.5	3.2	3.0	3.0	3.0	3.1	3.0	3.0
50	50	0.2	5.9	5.5	5.9	5.8	5.8	5.8	5.3
50	62.5	0.8	2.7	2.7	2.7	2.8	2.9	2.7	2.6
50	62.5	0.5	3.6	3.3	3.2	3.5	3.7	3.6	3.2
50	62.5	0.2	6.2	6.6	6.7	6.5	6.4	6.2	6.0
Sum			37.3	36.2	36.7	36.7	37.3	36.7	35.1

Table 4
Computational results for $n = 100$ and $q = 0.4n$

n	l	d	Greedy	VNS1	VNS2	VNS3	SA	RTS	Pilot
100	25	0.8	1.0	1.0	1.0	1.0	1.0	1.0	1.0
100	25	0.5	1.9	1.9	1.9	1.9	1.9	1.9	1.9
100	25	0.2	3.1	3.0	3.0	3.0	3.0	3.0	3.0
100	50	0.8	2.0	2.0	2.0	2.0	2.0	2.0	2.0
100	50	0.5	2.4	2.2	2.2	2.2	2.4	2.3	2.2
100	50	0.2	4.8	4.4	4.5	4.8	4.4	4.6	4.3
100	100	0.8	3.0	3.0	3.0	3.0	3.0	3.0	3.0
100	100	0.5	3.9	3.6	3.6	3.6	3.8	3.9	3.6
100	100	0.2	6.6	6.9	7.6	7.0	7.4	6.6	6.5
100	125	0.8	3.0	3.0	3.0	3.0	3.4	3.0	3.0
100	125	0.5	4.1	4.1	4.2	4.3	4.9	4.1	4.0
100	125	0.2	7.6	8.1	9.2	8.2	8.1	7.6	7.0
Sum			43.4	43.2	45.2	44.0	45.3	43.0	41.5

and 32 of the cases with $q = 0.4n$ the objective turned out to be larger than 5, i.e., 6, 7, or 8). In this sense, a random neighbor selection within the SA and the VNS implementations may already lead to an unfavorable objective function value that is difficult to be overcome which explains the few cases where the SA results are even worse than those of the greedy approach.

For RTS the approach first mimics the behaviour of a steepest descent like the greedy heuristic. Based on the way infeasibilities are penalized, the method usually stays within the feasible region so that the method may be caught within some basin of attraction related to the first local optimum found. That is, the RTS does not really work as expected, since in most cases (about 95%) the best results have been obtained within the first second of the computation. After that the method did not find improvements quite often even if they would have been possible.

The computations for our methods have been made on a Pentium IV 1.8 GHz. The termination criteria for the different methods follow the descriptions given above. The RTS is terminated after a time limit of 10 seconds for instances with $n = 50$ and after 40 seconds for $n = 100$. In general the computational times are moderately increasing for decreasing values of d, they are also increasing for increasing values of q and l, and they are considerably increasing for an increasing number of nodes. While RTS has a given time limit, computational times for the other methods tend to be below those numbers for larger values of d for all methods while they become slightly larger than those for RTS in case of the VNS implementations for $d = 0.2$. Computational times for the VNS implementations mainly depend on graph density: the more sparse the graph the larger the times. The computational times for the pilot method mainly depend on the number of nodes and the value of l. If l increases then the times for the pilot method may easily become considerably larger than those of SA and RTS but also larger than those of the VNS. Detailed computational times are reported in Table 5 for the largest instances to get a feeling about the general behaviour of our methods.

Table 5
Computational times [s] for $n = 100$ and $q = 0.4n$

n	l	d	VNS1	VNS2	VNS3	SA	RTS	Pilot
100	25	0.8	0.4	0.2	4.6	11.1	40.0	1.0
100	25	0.5	0.6	0.6	2.8	9.7	40.0	1.2
100	25	0.2	19.2	23.4	46.0	6.6	40.0	1.4
100	50	0.8	1.1	1.2	7.0	22.1	40.1	6.5
100	50	0.5	12.0	5.2	28.2	17.4	40.0	6.1
100	50	0.2	49.9	81.6	77.0	12.3	40.0	9.1
100	100	0.8	15.3	10.9	57.8	43.5	40.1	43.3
100	100	0.5	44.8	101.9	50.3	33.7	40.0	43.0
100	100	0.2	72.0	128.2	96.2	21.7	40.0	63.5
100	125	0.8	29.0	66.0	36.8	54.0	40.1	73.1
100	125	0.5	50.5	75.5	53.4	40.5	40.1	76.7
100	125	0.2	94.3	174.8	128.0	27.3	40.0	115.1

We should note that a detailed analysis of the results reveals that the pilot method usually does not need as much time as shown to find the indicated solutions, since in almost all cases the best result has been obtained in the first few seconds of the computations. This gives a strong hint that a small evaluation depth (see [18]) may be used to reduce the computation times without discarding solution quality.

6. Conclusions and further research

In this paper we have considered a generalization of the minimum labeling spanning tree problem to the case where not necessarily all but only a subset of required nodes need to be spanned. Common metaheuristics have successfully been applied to this generalization and the results are in line with our expectation gained from experimentation with the original labeling spanning tree problem. The most visible result is that the pilot method outperforms the other approaches with respect to solution quality while the computation times of the pilot method can be considerably larger than those of reactive tabu search or simulated annealing especially for larger problem instances. The computation times of our implementations for the pilot method and the variable neighborhood search are somewhat comparable with some exceptions for smaller densities of the given graphs where the pilot method may be faster. This motivates one direction of our further research consisting in the combination of the two metaheuristics that seem to behave better, the pilot method and the VNS1 [4]. Moreover, the results have been obtained for one generalization of the labeling spanning tree problem and future research refers also to extending those ideas to other generalizations such as the one also proposed in this paper considering additional budget constraints. Moreover, allowing for more than one color assigned to each edge poses an interesting case motivated by some applications.

Another step in our research refers to developing various mathematical programming formulations for the MLST as well as the MLSteiner to obtain optimal solutions to better judge on the quality of our heuristic solutions at least for small and moderately sized problem instances.

References

[1] R. Battiti, "Reactive search: toward self-tuning heuristics", in *Modern Heuristic Search Methods*, V. J. Rayward-Smith, I. H. Osman, C. R. Reeves, and G. D. Smith, Eds. Chichester: Wiley, 1996, pp. 61–83.

[2] H. Broersma and X. Li, "Spanning trees with many or few colors in edge-colored graphs", *Discus. Math. Graph Theory*, vol. 17, pp. 259–269, 1997.

[3] T. Brüggemann, J. Monnot, and G. J. Woeginger, "Local search for the minimum label spanning tree problem with bounded color classes", *Oper. Res. Lett.*, vol. 31, pp. 195–201, 2003.

[4] R. Cerulli, A. Fink, M. Gentili, and S. Voß, "Applications of the pilot method and VNS to hard modifications of the minimum spanning tree problem", in *Mini Euro Conf. VNS*, Tenerife, Spain, 2005.

[5] R. Cerulli, A. Fink, M. Gentili, and S. Voß, "Metaheuristics comparison for the minimum labelling spanning tree problem", in *The Next Wave on Computing, Optimization, and Decision Technologies*, B. L. Golden, S. Raghavan, and E. A. Wasil, Eds. New York: Springer, 2005, pp. 93–106.

[6] R.-S. Chang and S.-J. Leu, "The minimum labeling spanning trees", *Inform. Proces. Lett.*, vol. 63, pp. 277–282, 1997.

[7] C. W. Duin and S. Voß, "The pilot method: A strategy for heuristic repetition with application to the Steiner problem in graphs", *Networks*, vol. 34, pp. 181–191, 1999.

[8] A. Fink, G. Schneidereit, and S. Voß, "Solving general ring network design problems by metaheuristics", in *Computing Tools for Modeling, Optimization and Simulation*, M. Laguna and J. L. González Velarde, Eds. Boston: Kluwer, 2000, pp. 91–113.

[9] F. Glover and M. Laguna, *Tabu Search*. Boston: Kluwer, 1997.

[10] F. K. Hwang and D. S. Richards, "Steiner tree problems", *Networks*, vol. 22, pp. 55–89, 1992.

[11] F. K. Hwang, D. S. Richards, and P. Winter, *The Steiner Tree Problem*. Amsterdam: North-Holland, 1992.

[12] D. S. Johnson, C. R. Aragon, L. A. McGeoch, and C. Schevon, "Optimization by simulated annealing: an experimental evaluation", Part I, "Graph partitioning", *Oper. Res.*, vol. 37, pp. 865–892, 1989.

[13] S. Kirkpatrick, C. D. Gelatt Jr., and M. P. Vecchi, "Optimization by simulated annealing", *Science*, vol. 220, pp. 671–680, 1983.

[14] S. O. Krumke and H.-C. Wirth, "On the minimum label spanning tree problem", *Inform. Proces. Lett.*, vol. 66, pp. 81–85, 1998.

[15] N. Mladenović and P. Hansen, "Variable neighbourhood search", *Comput. Oper. Res.*, vol. 24, pp. 1097–1100, 1997.

[16] S. Voß, "Modern heuristic search methods for the Steiner tree problem in graphs", in *Advances in Steiner Trees*, D.-Z. Du, J. M. Smith, and J. H. Rubinstein, Eds. Boston: Kluwer, 2000, pp. 283–323.

[17] S. Voß, "Steiner tree problems in telecommunications", in *Handbook of Optimization in Telecommunications*, M. Resende and P. M. Pardalos, Eds. New York: Springer, 2006, pp. 459–492.

[18] S. Voß, A. Fink, and C. Duin, "Looking ahead with the pilot method", *Ann. Oper. Res.*, vol. 136, pp. 285–302, 2004.

[19] Y. Wan, G. Chen, and Y. Xu, "A note on the minimum label spanning tree", *Inform. Proces. Lett.*, vol. 84, pp. 99–101, 2002.

[20] P. Winter, "Steiner problem in networks: a survey", *Networks*, vol. 17, pp. 129–167, 1987.

[21] Y. Xiong, B. Golden, and E. Wasil, "A one-parameter genetic algorithm for the minimum labeling spanning tree problem". Tech. Rep., University of Maryland, 2003.

[22] Y. Xiong, B. Golden, and E. Wasil, "Improved metaheuristics for the minimum labeling spanning tree problem". Tech. Rep., University of Maryland, 2005 (also in *Metaheur. Int. Conf.*, Vienna, Austria, 2005).

[23] Y. Xiong, B. Golden, and E. Wasil, "Worst-case behavior of the MVCA heuristic for the minimum labeling spanning tree problem", *Oper. Res. Lett.*, vol. 33, pp. 77–80, 2005.

Raffaele Cerulli is currently Associate Professor in operations research at the Department of Mathematics and Computer Science of the University of Salerno, Italy. He holds laurea degree in computer science from University of Salerno. His current research interests are in combinatorial optimization problems arising from real applications in the field of telecommunications and logistics. In particular, he focused on network flow problems as well as covering problems on graphs. He has several papers in various journals and he was Director of the International School on "Pattern Analysis" as well as the organizer of workshops on Graph Theory in Italy.

e-mail: raffaele@unisa.it
Department of Mathematics and Computer Science
University of Salerno
Via Ponte Don Melillo
84084, Fisciano (Salerno), Italy

Andreas Fink is Professor and the Chair of Business Administration, in particular Information Systems, at the Helmut-Schmidt-University in Hamburg, Germany. He holds diploma degrees in business administration and computer science from the University of Technology Darmstadt and the Ph.D. in economics from the University of Technology Braunschweig. His research is mainly concerned with the use of information technology to support decision-making in fields such as telecommunications, logistics, and supply chain management. His publications have appeared in various journals.

e-mail: andreas.fink@hsu-hamburg.de
Department of Economics
Helmut-Schmidt-University / UniBw Hamburg
Holstenhofweg 85
22043 Hamburg, Germany

Monica Gentili is currently Assistant Professor in operations research at the Department of Mathematics and Computer Science of the University of Salerno, Italy. She received the laurea degree in statistics in 1998 and the Ph.D. in operations research in 2003 at the University of Rome "La Sapienza". Her current research interests are concentrated on combinatorial optimization problems, mainly on mathematical models and algorithm design for planning and control of traffic flows on networks as well as covering problems on graphs. They include sensor location problems on networks, routing and distribution problems, graph theory.

e-mail: mgentili@unisa.it
Department of Mathematics and Computer Science
University of Salerno
Via Ponte Don Melillo
84084, Fisciano (Salerno), Italy

Stefan Voß – for biography, see this issue, p. 20.