J Comb Optim (2016) 31:1461–1478
DOI 10.1007/s10878-015-9834-5

Heuristics for the network design problem with connectivity requirements

Roman E. Shangin · Panos Pardalos

Published online: 17 February 2015

Abstract We consider the NP-complete problem of finding a spanning k-tree of minimum weight in a complete weighted graph. This problem has a number of applications in designing reliable backbone telecommunication networks. We propose effective algorithms based on a greedy strategy and several variable neighborhood search metaheuristics. We also develop an integer linear programming model for calculating a lower bound. Preliminary numerical experiments using random and real-word data sets are reported to show the effectiveness of our approach. In addition, we compare our approach with known metaheuristics.

Keywords K-trees · Robust networks · Network design · Heuristics · Variable neighborhood search

1 Introduction

Nowadays, due to a significant increase in scales and complexity of telecommunication infrastructures, one of the most important problems is designing telecommunication networks of minimal costs with a given level of network reliability (Thai and Pardalos 2011; Pardalos and Thai 2011; Dinh et al. 2012). Traditionally, this problem is considered in two variants. The first one, when a network topology is not specified and it is computed in the process of solving the problem (Haidine 2013; Elshqeirat et al.

R. E. Shangin (✉)
Department of Computational Mathematics and Informatics, South Ural State University, Cheliabinsk, Russia
e-mail: shanginre@gmail.com

P. Pardalos
Department of Industrial and Systems Engineering, University of Florida, Gainesville, USA
e-mail: pardalos@ufl.edu

2013; Dengiza et al. 2010; Johnston et al. 2013). The second one, when the network topology is given (Konak and Smith 2011; Khandekara et al. 2013; Simonetti et al. 2011; Alvarez-Mirandaa et al. 2013; Bansal et al. 2013). In the second variant, connectivity requirements of a network are a kind of surrogate for reliability (Magnanti and Raghavan 2005). Researches in this field have led to the appearance of the concept of isolated failure immune (IFI) networks (Farley 1981) in the 80s of the last century. Such networks remain connected even in the presence of a large number of failures. Specifically, IFI networks work with three types of failures (Farley 1981; Candia and Bravo 2002):

- $[i, j]$ and $[p, q]$ are two *line isolated failures* if $[i, j]$ and $[p, q]$ are not incident to a common node;
- i and j are two *node isolated failures* if i and j are not connected;
- a *line failure* $[i, j]$ and a *node failure* p are isolated if $[i, j]$ is not incident to p or to a node connected to p.

A set of failures is isolated if failures in the set are pairwise isolated. The network is IFI if it remains connected as long as network failures are isolated. In Farley (1981) it is proved that 2-trees are minimal (with respect to edge inclusion) IFI networks. Many problems of designing IFI networks can be formulated as a *Minimum Spanning k-Tree Problem* (MSkT), which generalizes the classical problem, the *Minimum Spanning Tree Problem* (MST) (Prim 1957).

The mathematical formulation of the MSkT problem is as follows. We are given a complete weighted undirected graph $G = (V, E)$, where V refers to a set of nodes (telecommunication stations) and E refers to a set of edges among $n = |V|$ nodes (possible links). The expense of each edge $[i, j]$ between vertices i and j is quantified as a nonnegative value $w(i, j)$. We are given a positive integer constant k. Let $T(G)$ be the set of all spanning k-trees in the graph G, where a spanning k-tree is a k-tree that contains all vertices and some edges of the graph G. Let $w(T)$ be the weight of edges of the spanning k-tree $T \in T(G)$. It is required to find the spanning k-tree T^* of minimum weight in the complete weighted graph G, that is $T^* = \arg\min_{T \in T(G)}\{w(T)\}$.

In Bern (1987), Cai and Maffray (1992) it is proved that the MSkT problem when $k \geq 2$ is NP-complete. In Bern (1987) it is proved that the cardinality of the set of feasible solutions of the MSkT problem equals to $n!(n + k - 1)!/k!$. Thus, the time complexity of an exhaustive search is $O(n^{2n}/k^k)$. Also in Bern (1987) there has been proposed a nonpolynomial exact algorithm based on dynamic programming, which has a significantly lower complexity $O(n^{k+1}3^n)$ than the exhaustive search. In Bern (1987) there has been proved inapproximability of the MS2T problem (i.e. where $k = 2$). In Beck and Candia (1993), Beck et al. (1993), Candia and Bravo (2002), Beltran and Skorin-Kapov (1993), Cai (1996), Beck and Candia (2000), Wald and Colbourn (1983) there have been proposed effective heuristics and metaheuristics for solving the MS2T. Although the MS2T is well studied, nevertheless, there is little attention given in the literature to the development of algorithms for solving the MSkT, where $k > 2$. The main purpose of this work is to develop effective heuristics and metaheuristics for solving the MSkT problem (i.e. where $k \geq 2$) on a complete weighted graph.

The plan of this paper is as follows. In Sect. 2, we give the definition of a k-tree and several of its basic properties. In Sect. 3, we propose heuristics that use a greedy

strategy. In Sect. 4, we propose variable neighborhood search metaheuristics. In Sect. 5, we present an integer linear program for computing a lower bound for the MSkT. In Sect. 6, we present results of computational experiments. In Sect. 7 we summarize our findings and discuss directions for further research.

2 Definition and basic properties of k-trees

Here is a well-known definition of a k-tree.

Definition (Rose 1974) A k-tree is a member of a class of undirected graphs defined recursively as follows: a complete graph with k vertices is a k-tree; if T is a k-tree with n vertices, then a new k-tree with $n+1$ vertices is formed by creating a new vertex v and adding edges between v and every vertex of an existing k-clique (clique with k vertices) in T.

The class of k-trees was introduced in the 70s of the last century by D. Rose. Besides using trees in the networks design, wide interest in the study of such graphs is caused by the fact that some NP-hard combinatorial problems are solvable in polynomial time when they are restricted to k-trees. Today there are polynomial time algorithms for solving some *classical problems of graph theory* on k-trees for example, maximum clique problem, minimum dominating set problem, Steiner tree problem, etc (Garey and Johnson 1976). Polynomial time algorithms for many *location problems* on k-trees are also known: simple location problem (Granot and Skorin-Kapov 1994), p-median and p-center problems (Shi 2008), etc.

If T is a k-tree with n vertices, then

1. All maximal cliques in T have size equals to k;
2. The size of a minimal separator (disconnecting set) in T equals to k;
3. The number of edges of the k-tree T equals to $nk - \frac{k(k+1)}{2}$;
4. The number of $(k+1)$-cliques in T equals to $n-k$;
5. The number of k-cliques in T equals to $(n-k)k+1$;
6. For all nonadjacent vertices $i, j \in V$, there exist exactly k vertex-disjoint paths.

The recursive definition of k-trees implies that they have the *perfect elimination order* property (Golumbic 1980). That is, for a k-tree T with n vertices, there exists an ordering of the set V, say $v_1, v_2, \ldots, v_j, \ldots, v_n$, such that in the node-induced subgraph $V \setminus \{v_j : j = 1, 2, \ldots, i-1\}$, where $i \geq 1$, nodes adjacent to v_i form a k-clique. A detailed study of k-trees properties is presented, for example, in Golumbic (1980), Rose (1970).

3 Heuristics based on greedy strategy

In this section, we propose greedy algorithms, which compute a feasible approximate solution for the MSkT. These algorithms will be used both for solving the MSkT problem and for computing a starting solution in metaheuristics.

3.1 Algorithm GREEDY

We propose the algorithm GREEDY which uses the idea of the well-known Prim's algorithm for the MST problem (Prim 1957). Let $W_i = (V_i', E_i')$ be a complete graph with i nodes. Let $T_i = (V_i, E_i)$ be a k-tree with i nodes and $K(T_i)$ is a set of all k-cliques in T_i.

ALGORITHM GREEDY

INPUT: Complete graph $G = (V, E)$ with positive edge weights. Integer $k \geq 2$.
OUTPUT: Spanning k-tree T_{opt} in G.

STEP 1 Find the minimum weight edge $[l, m]^*$ in G. Construct the graph W_2, where $V_2' = \{l, m\}$ and $E_2' = [l, m]^*$. Let $i = 2$.

STEP 2 (*Compute the "starting" clique of size $k+1$ in G*)
Find the vertex $m^* \in V \setminus V_i'$ for which the total weight $\sum_{j \in V_i'} \omega(m^*, j)$ of edges which connect this vertex m^* with nodes of the complete graph W_i is minimal. Construct the complete graph W_{i+1} by including the computed vertex m^* and the set of edges $\{[m^*, j] : j \in V_i'\}$ in the graph W_i. Let $i = i + 1$.
IF $|V_i'| = k + 1$, THEN $T_i = W_i$ and go to step 3, ELSE go to step 2.

STEP 3 (*Compute the k-tree T_{i+1} with $i+1$ vertices*)
Find the vertex $m^* \in V \setminus V_i$ and the k-clique $K^* \in K(T_i)$ for which the total weight $\sum_{j \in K^*} w(m^*, j)$ of edges which connect this vertex m^* with this k-clique K^* is minimal. Construct the new k-tree T_{i+1} by including the computed vertex m^* and the set of edges $\{[m^*, j] : j \in K^*\}$ in the k-tree T_i. Let $i = i + 1$.
IF $|V_i| = n$, THEN $T_{opt} = T_i$ and **STOP** the algorithm, ELSE go to step 3.

Proposition 1 *The time complexity of the algorithm* GREEDY *is* $O(n^3 k)$.

Proof At step 1 for finding the cheapest edge in graph G one needs $O(|E|)$ operations. Step 2 can be implemented in $O((k-1) \cdot n)$ operations because $O(n)$ operations are necessary to compute the new vertex $m^* \in V \setminus V_i'$ at every iteration of step 2, and the number of such iterations at step 2 equals to $k - 1$. At every iteration of step 3 one needs $O((n-k)^2 \cdot k)$ operations because there are $(n-k) \cdot k$ cliques of the size k in the k-tree T_i, and the number of all possible variants of the new vertex $m^* \in V \setminus V_i$ at each iteration of step 3 does not exceed $n - k$. As far as the number of iterations performed at step 3 is $n - k - 1$, then at step 3 one requires $O((n-k)^3 \cdot k)$ operations. Based on this, the time complexity of the algorithm GREEDY is $O(n^3 k)$. □

Despite the fact that the idea of the algorithm GREEDY is simple and natural, this heuristic computes approximate solutions of good quality (see Sect. 6.1). Note that, the GREEDY is a modification of the known heuristic proposed by Beck and Candia (2000). The basic difference between our algorithm and the known heuristic is in the method of computing the "starting" clique. In our algorithm it is computed in order to minimize the weight of edges in W_i and in Beck and Candia's heuristic it is chosen randomly. Figure 1 shows the ways of constructing solutions for the MSkT problem by the GREEDY heuristic.

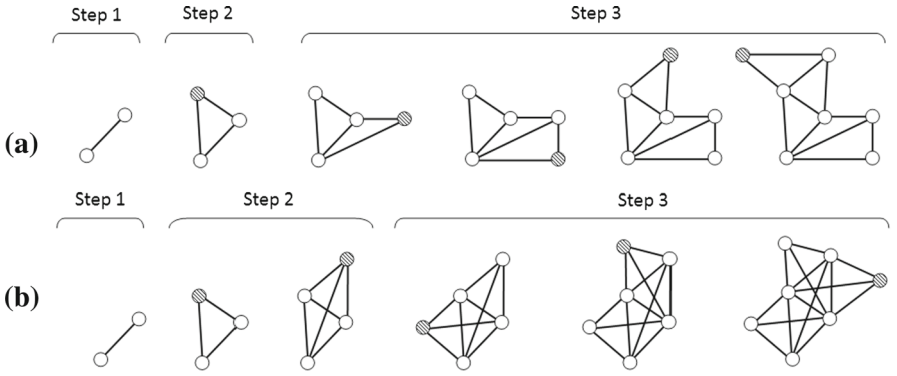

Fig. 1 Construction of the solutions by the GREEDY heuristic for: **a** MS2T; **b** MS3T

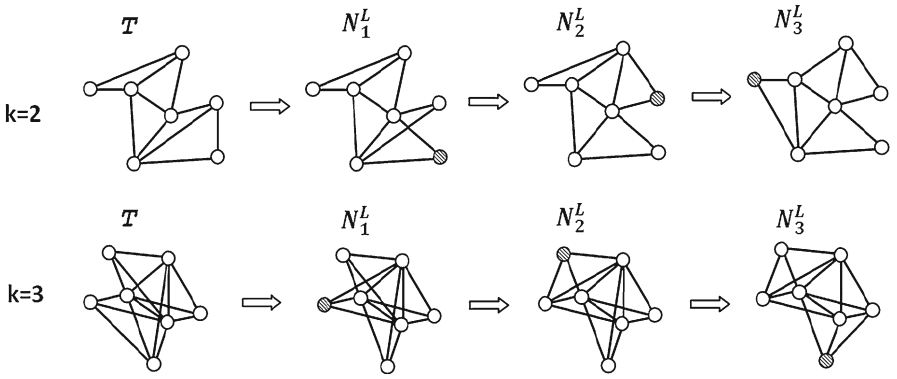

Fig. 2 Elements of the neighborhoods $N_l^L(T)$ with $l = 1, 2, 3$ for 2-tree and 3-tree

3.2 Neighborhoods

Let C_T be a set of all cliques of a size $k+1$ in a k-tree T. The *neighborhood* $N_l^L(T)$ of the solution T is a set of spanning k-trees that differ from T by exactly l cliques of a size $k+1$, that is $N_l^L(T) = \{T' : |C_{T'} \setminus C_T| = l\}$. The element T' of the neighborhood $N_l^L(T)$ can be built from T by l-fold repetition of the operations: delete k edges connecting some simplicial vertex j' (k-leaf) with vertices of some k-clique K and then add k new edges connecting the vertex j' with another k-clique K', where $K' \neq K$. The cardinality of the neighborhood N_l^L does not exceed $O(n^{l+1}k^l)$ because the number of all possible connections of a k-leaf with other k-cliques is at most nk and the number of k-leaves does not exceed n. Figure 2 shows elements of the neighborhoods $N_l^L(T)$ for 2-tree and 3-tree.

It is quite obvious that the search within the neighborhood N_l^L for large values of l is not justified from the computational point of view, despite the fact that it very often leads to really good approximate solutions. This drawback of N_l^L can be eliminated by using *the Kernighan–Lin neighborhood* $\mathcal{K}_l^L$, which includes the neighborhood N_1^L

and is part of N_l^L. Its main idea is to find the best solution in N_l^L by using a greedy procedure. The Kernighan–Lin neighborhood of T is defined by the following steps:

Step 0 Let $T_0 = T$.

Step 1 Find the solution T' in $N_1^L(T_0)$ of minimal weight. Let $T_0 = T'$.

Step 2 Repeat step 1 l times such that we do not consider vertices that have already been used in the previous iterations.

Obviously, $O(ln^2k)$ operations are required for computing the neighborhood $\mathcal{K}_l^L$ because $O(n^2k)$ operations are required to build the neighborhood N_1^L, and $\mathcal{K}_l^L$ contains l such neighborhoods.

3.3 Modifications of the Greedy heuristic

We propose several modifications of the Greedy, aimed at improving its accuracy.

The first modification Greedy^1 is to combine the heuristic Greedy with a local descent algorithm. In this modification at every iteration $i = k+2, \ldots, n$ the deterministic steepest local descent with the neighborhood N_1^L will be applied to improve the computed solution T_{i+1}. If the local descent algorithm is able to find a lower-cost solution, then solution T_{i+1} is replaced by the new computed solution, otherwise the solution T_{i+1} does not change. The time complexity of this modification is also $O(n^3k)$ because the local descent with N_1^L requires no more than $O(n^2k)$ operations at every iteration of step 3.

The second modification Greedy^2 is as follows. We sort edges from G in increasing order based on their weights and select the first n edges. For every selected edge $[x, y]$ we construct a spanning k-tree using the Greedy heuristic, where we select the edge $[x, y]$ at step 1. A solution of this algorithm is a spanning k-tree of minimum weight over all computed k-trees. The time complexity of this modification is $O(n^4k)$ because the number of constructed k-trees by this algorithm equals to n.

The third modification Greedy^3 combines ideas of two previous modifications. In the algorithm for every selected edge we construct a spanning k-tree using the Greedy^1 algorithm. Then we find a spanning k-tree of minimum weight over all computed k-trees. Essentially, the computational complexity of this modification is also $O(n^4k)$.

4 Variable neighborhood search metaheuristics

The variable neighborhood search metaheuristic was proposed by Hansen et al. (2010). This approach uses several neighborhoods and changes them systematically that allows to escape from local optima and find better and better solutions.

4.1 Basic scheme of variable neighborhood search

The variable neighborhood search algorithm can be implemented in many ways, such as deterministic (VND), probabilistic (RVNS) or mixed (VNS). In this paper, we will focus on the mixed scheme because it combines strengths of both schemes.

There are two phases in the VNS: *the shaking phase*, thanks to which the algorithm escapes from local optima, and *the local search phase*, where the iterative improvement is carried out from solutions generated at the shaking phase. Denote by $N_t^S, t = 1, 2, \ldots, t_m$ neighborhoods used at the shaking phase and let $N_l^L, l = 1, 2, \ldots, l_m$ be neighborhoods used at the local search phase. Let I_m be a maximal number of iterations at every t.

ALGORITHM VNS

INPUT: Complete graph $G = (V, E)$. Integer $k \geq 2$. Starting solution T_0. Neighborhoods $N_t^S, t = 1, 2, \ldots, t_m$ and $N_l^L, l = 1, 2, \ldots, l_m$. Integer I_m.
OUTPUT: Spanning k-tree T_{opt} in G.

Let $t = 1$, $I = 1$ and $T^* = T_0$.

STEP 1 Repeat steps 1.1-1.2 while $w(T') \geq w(T^*)$ and $I \leq I_m$.

STEP 1.1 (*shaking*) Generate a solution T' at random from the neighborhood $N_t^S(T^*)$.

Let $l = 1$.

STEP 1.2 (*local search*) Find the best solution T'' in the neighborhood $N_l^L(T')$.

- IF $w(T'') < w(T')$, THEN let $T' = T''$ and $l = 1$, ELSE let $l = l + 1$.
- IF $l \leq l_m$, THEN repeat step 1.2, ELSE $\lambda = \lambda + 1$ and go to step 1.1.

STEP 2 Find the best solution $\hat{T}$ computed at step 1 for a given t.

- IF $w(\hat{T}) < w(T^*)$, THEN update the record $T^* = \hat{T}$ and $t = 1$, ELSE $t = t + 1$.
- IF $t \leq t_m$, THEN let $\lambda = 1$ and go to step 1, ELSE **STOP**.

As the starting solution, we will use the solution found by the GREEDY^3 because of its high accuracy and small computational time costs (see Sect. 6.1).

4.2 Neighborhoods

Denote by $S = \{1, 2, \ldots, i, \ldots, j, \ldots, n\}$ a sequence of vertices of a spanning k-tree, where the vertices are ordered in accordance with the perfect order elimination property (see Sect. 2). Note that for any k-tree we can compute the sequence of vertices S in linear time (Golumbic 1980). Obviously, from some sequence S we can construct a spanning k-tree by the following modification of the GREEDY algorithm. In this modification at the iteration i of step 3 we include the vertex with number i from the sequence S and the set of edges $\{[i, j] : j \in K^*\}$ of minimal weight. Naturally, the time complexity of this modification is $O(n^2k)$. Let $S^{(t)}$ be a set of sequences obtained from the sequence S by sequential permutation of t pairs of its elements. For example, the set $S' = \{\underline{1}, 6, 3, 2, \underline{4}, 5\}$ is obtained from $S = \{\underline{4}, 6, 3, 2, \underline{1}, 5\}$ by permutation of the one pair of vertices and hence $S' \in S^{(1)}$.

The neighborhood $N_t^S(T)$ of the solution T is a set of spanning k-trees constructed from sequences belonging to the set $S^{(t)}$, where S is the sequence of vertices corresponding to the k-tree T. Naturally, the cardinality of the neighborhood N_t^S is at most $O(n^{2t})$. Note that the idea of using such a neighborhood was partially used in Candia and Bravo (2002), Beltran and Skorin-Kapov (1993).

4.3 Modifications of the basic scheme

Denote by VNS(t_m, l_m, I_m) the algorithm VNS with the maximum neighborhoods $N^S_{t_m}$ and $N^L_{l_m}$ and the maximum number of iterations I_m, performed for every t.

Proposition 2 *The time complexity of the algorithm VNS(t_m, l_m, I_m) does not exceed* $O(t_m I_m n^{l_m+1} k^{l_m})$.

Proof The sequence S' is chosen randomly from the set $S^{(t)}$ that requires no more than $O(t_m)$ operations. Computing a solution T' from S' requires $O(n^2k)$ time and a number of repetitions of this step equals to $t_m I_m$. Hence, step 1.1 of the VNS requires $O(t_m I_m n^2 k)$ operations. Step 1.2 requires $O(t_m I_m n^{l_m+1} k^{l_m})$ operations because the cardinality of N^L_l does not exceed $O(n^{l_m+1} k^{l_m})$, and a number of repetitions of step 1.2 equals to $t_m I_m$. Hence, the time complexity of the algorithm VNS(t_m, l_m, I_m) does not exceed $O(t_m I_m n^{l_m+1} k^{l_m})$. □

Next, we propose several modifications of the VNS algorithm.

The first modification VNS$^1(t_m, l_m, I_m)$ of the VNS consists in changing the way of choosing a solution from N^S_t at step 1.1 (shaking phase). In this modification, the new solution is not selected from N^S_t at random, but by analogy with *the simulated annealing approach* we select the solution, or with lower costs then record, or with larger costs then record, but with a probability depending on its costs. Based on this, step 1.1 of the VNS^1 is as follows:

Step 1.1 Select T at random from the neighborhood $N^S_t(T^*)$. If $w(T) < w(T^*)$, then let $T' = T$, else let $T' = T$ with the probability $e^{-\frac{\Delta w \cdot \gamma}{\lambda}}$, where $\Delta w = \frac{w(T)-w(T^*)}{w(T^*)}$ and $\gamma = 1, 2, ...$ is a sensitivity coefficient of the probability from the solution's costs. If T' is selected, then go to step 1.2, else repeat step 1.1.

This method of selecting solutions from the neighborhood $N^S_t(T^*)$ has the following meaning. The smaller the costs of the solution are, the more likely its choice is. The longer the algorithm can not improve the record, the lower the costs of the solution T affect the probability of its selecting. The time complexity of the algorithm VNS^1 equals to the complexity of the VNS because this modification requires $O(1)$ repetitions of step 1.1, while the solution T' is not selected. The numerical experiments showed that the algorithm achieves the highest accuracy when $\gamma \in [4, 7]$. Based on this, we assume that the value γ equals to 5.

The second modification VNS$^2(t_m, l_m, I_m)$ of the basic scheme consists in using the Kernighan–Lin neighborhood at the local search phase. The time complexity of this algorithm is $O(t_m I_m l_m n^2 k)$ because the cardinality of $\mathcal{K}^L_{l_m}$ equals to $O(l_m n^2 k)$.

The third modification VNS$^3(t_m, l_m, I_m)$ contains ideas of the previous two modifications. The time complexity of the third modification also equals to $O(t_m I_m l_m n^2 k)$.

5 Lower bound for the MSkT problem

We present an integer linear programming (ILP) model for calculating the lower bound of the MSkT problem. This model is based on well-known properties of a k-tree which are partially given in statement 1. We have the vertex set $V = 1, 2, \ldots, n$. For any $i, j \in V, i < j$ the boolean variable x_{ij} is defined, where $x_{ij} = 1$ if the edge $[i, j]$ is included in the solution, and $x_{ij} = 0$ otherwise. The ILP model for the lower bound is as follows:

$$F_{LB} = \min_{x} \left\{ \sum_{i,j \in V:\, i<j} w(i, j) x_{ij} \right\} \tag{1}$$

subject to

$$\sum_{i,j \in V:\, i<j} x_{ij} = nk - \frac{k(k+1)}{2}, \tag{2}$$

$$\sum_{j \in V:\, i<j} x_{ij} + \sum_{j \in V:\, i>j} x_{ji} \geq k; \qquad (\forall i \in V), \tag{3}$$

$$\sum_{i,j \in S:\, i<j} x_{ij} \leq |S|k - \frac{k(k+1)}{2}; \qquad (\forall S \subseteq V, |S| \geq k), \tag{4}$$

$$\sum_{i \in S, j \in V \setminus S:\, i<j} x_{ij} + \sum_{i \in S, j \in V \setminus S:\, i>j} x_{ji} \geq \frac{k(k+1)}{2}; \qquad (\forall S \subseteq V, |S| \geq k), \tag{5}$$

$$x_{ij} \in \{0, 1\},\ i < j,\ i, j \in V. \tag{6}$$

Constraint (2) fixes the total number of edges to $nk - k(k+1)/2$. Constraints (3) say that every vertex of a k-tree must be connected by at least k vertices. Constraints (4) say that a number of edges in a subgraph induced by a set of vertices $S, |S| \geq k$ does not exceed the number of edges in a k-tree with $|S|$ vertices. Constraints (5) say that a set of vertices $S, |S| \geq k$ of some subgraph in a k-tree is connected with other vertices of a k-tree by at least $k(k+1)/2$ edges. Note that this ILP model partially uses the idea of computing a lower bound for the MS2T problem (where $k = 2$) (Beltran and Skorin-Kapov 1993), and the idea of computing a lower bound for the minimum spanning 2-connected graph problem (Grotschel and Monma 1992).

Obviously, this ILP problem has an exponential number of constraints and it is difficult to handle directly. We propose the algorithm LB for computing the lower bound which has a significantly lower time complexity than the model (1)–(6). The algorithm LB uses an idea of a sequential inclusion of constraints (4) and (5) for some $S \subseteq V$ in the model (1)–(3), (6). At every step of this algorithm we consider a relaxation of problem (1)–(6), and its objective function is also a lower bound for the MSkT problem.

ALGORITHM LB

INPUT: Complete graph $G = (V, E)$. Integer $k \geq 2$. ILP model (1)–(3), (6). Maximal number of iterations I_m.
OUTPUT: lower bound F_{LB} for the MSkT.

STEP 0 Compute the solution T of the ILP problem with constraints (2), (3) and (6). Let $T3 = T$ and $I = 1$.

STEP 1 In the graph $T3$ for every vertex $i \in V$ determine the set of vertices $N_{T3}(i)$ that includes the vertex i and all adjacent with i vertices in the graph $T3$. Add constraints

$$\sum_{i,j \in N_{T3}(i):\ i<j} x_{ij} \leq |N_{T3}(i)|k - \frac{k(k+1)}{2}; \qquad (\forall i \in V), \tag{7}$$

in the previous ILP model. Compute the solution $T1$ of the new ILP model. **STEP 2** If the graph $T1$ computed at step 1 is connected, then let $T2 = T1$ and go to step 3. If the graph $T1$ is disconnected, then compute the set C of its connected components. Let V_Y be a set of vertices of the connected component $Y \in C$. Add constraints

$$\sum_{i \in V_Y, j \in V \setminus V_Y:\ i<j} x_{ij} + \sum_{i \in V_Y, j \in V \setminus V_Y:\ i>j} x_{ji} \geq \frac{k(k+1)}{2}; \qquad (\forall Y \in C),$$

in the ILP model. Compute the solution $T2$ of the new ILP model.

STEP 3 Compute the set X of separators with capacity $< k$ in the graph $T2$. If the set X is empty, then let $T3 = T2$ and go to step 4. If X is not empty, then for every separator $Q \in X$ compute the set C_Q of connected components in the graph $T2$ obtained by removing the separator Q, $|Q| = q$. Add constraints

$$\sum_{i \in V_Y, j \in V':\ i<j} x_{ij} + \sum_{i \in V_Y, j \in V':\ i>j} x_{ji} \geq \frac{(k-q)(k-q+1)}{2}; \quad (\forall Q \in X, \forall Y \in C_Q), \tag{8}$$

in the ILP model, where $V' = (V \setminus Q) \setminus V_Y$. Compute the solution $T3$ of the new ILP.

STEP 4 IF the number of iterations I exceeds I_m OR $w(T3) = w(T1)$, THEN let $F_{LB} = w(T3)$ and **STOP**, ELSE $I = I + 1$ and go to step 1.

Constraints (7) say that a subgraph of a k-tree which is induced by a vertex $i \in V$ and vertices adjacent to i is also a k-tree (Rose 1970). Obviously, constraints (7) are a special case of constraints (4). Constraints (8) say that when we remove q, $q < k$ vertices that belong to a minimal separator Q in a k-tree, vertices of every subgraph $Y \in C_Q$, $|Y| \geq k$ will be connected with other vertices by at least $(k-q)(k-q+1)/2$ edges (Rose 1970).

Evidently, the number of separators of a size $< k$ in the graph $T2$ is at most n^{k-1}, and the cardinality of the set C_Q does not exceed nk. Hence, a number of constraints added to the ILP model at every iteration does not exceed n^k. The numerical experiments indicated that for problems MS2T and MS3T of dimension $|V| = 20$ at every iteration

Table 1 The time complexity of the proposed algorithms

Algorithm	Time complexity	Algorithm	Time complexity
GREEDY	$O(n^3k)$	VNS	$O(t_m I_m n^{l_m+1} k^{l_m})$
GREEDY^1	$O(n^3k)$	VNS^1	$O(t_m I_m n^{l_m+1} k^{l_m})$
GREEDY^2	$O(n^4k)$	VNS^2	$O(t_m I_m l_m n^2 k)$
GREEDY^3	$O(n^4k)$	VNS^3	$O(t_m I_m l_m n^2 k)$

the average number of added constraints was 17 and 26, respectively, and the average number of iterations was 6 and 8, respectively, when maximal number I_m was 15.

6 Experimental results

The numerical experiments were performed to compare the effectiveness of the proposed algorithms with known metaheuristics. By efficiency of an algorithm we understand its running time and accuracy. All algorithms were implemented in "MATLAB R2011a". To calculate the lower bound of the MSkT we used "IBM ILOG CPLEX Optimization Studio 12.2" (solving the ILP by the branch and bound algorithm). Calculations have been performed on a PC with the processor "Intel Core i7 2.6 GHz". The study of the efficiency of the algorithms was performed on two classes of instances:

- instances where weights of edges are Euclidean distances (class Euclidean);
- instances where weights of edges are generated at random with uniform distribution (class Uniform).

We also tested the algorithms on a real-world data of the European backbone IP-network provided by the "TeleGeography" consultancy (http://www.telegeography.com). Denote by $\bar{t}$ an average running time of an algorithm, sec.; $\bar{\varepsilon}$ – an average relative error of an algorithm, %; σ – standard deviation of the relative error. Note that, the value ε is computed by the formula $\varepsilon = (w_{alg} - w_{LB})/w_{LB} \cdot 100\,\%$, where w_{alg} is the weight of a spanning k-tree constructed by the algorithm, and w_{LB} is the lower bound computed by the algorithm LB.

Table 1 presents the time complexity of the proposed algorithms.

6.1 Analysis of the algorithms based on the greedy strategy

Figure 3 shows the results of the computational experiments to analyze the efficiency of the algorithm GREEDY and its modifications. The experiments were conducted on sets of instances, where every set includes 20 instances of the same dimension.

On the Uniform class the algorithm GREEDY^3 showed the best results in terms of accuracy, and values of its average error were less than 2.2 and 2.9 % for the problems MS2T and MS3T, respectively. For the problems MS2T and MS3T the algorithm GREEDY showed the worst results in terms of accuracy. Furthermore, there was a significant increase of its relative error by increasing dimension of the problem.

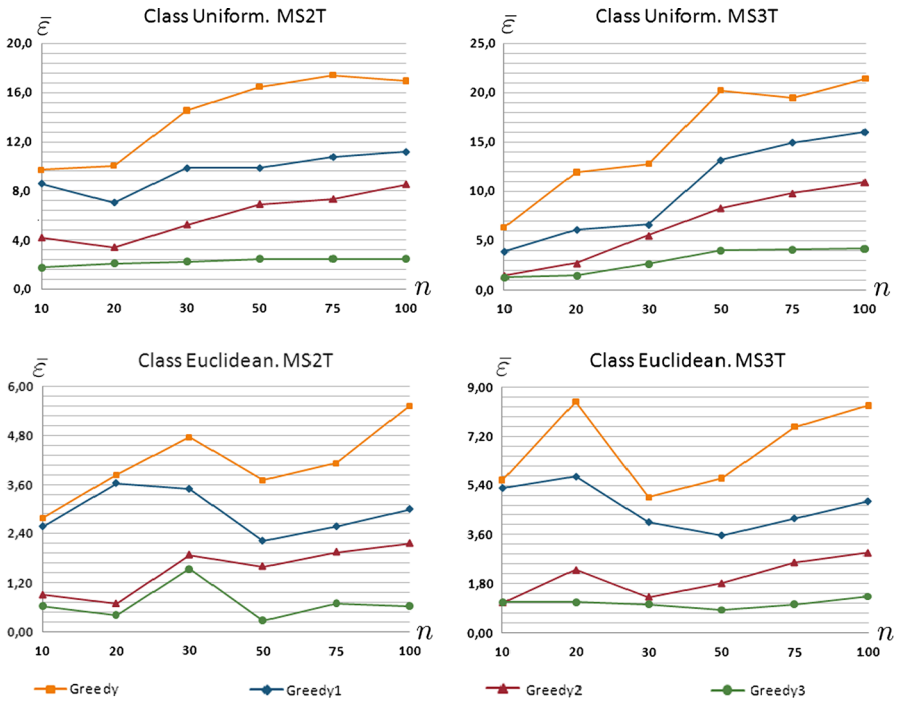

Fig. 3 The average relative errors of the algorithms based on the greedy strategy

On the Euclidean class errors of the greedy algorithms were much lower than on the Uniform class, that takes place, apparently, due to the triangle inequality at the matrices of distances. Despite this, GREEDY showed the worst results and GREEDY^3 showed the best results in terms of accuracy.

6.2 Analysis of the variable neighborhood search algorithms

Figure 4 illustrates the results of the experiments to analyze the efficiency of the variable neighborhood search algorithms.

On all sets of instances for Uniform and Euclidean classes errors of the algorithm VNS and all its modifications were less than 2 % that indicates high accuracy of such metaheuristics. The error in most cases was because it is not calculated with respect to an optimum but with respect to the lower bound, which is often less than the optimum. The algorithm VNS^1 showed the best results in terms of accuracy but its running time for instances of dimension $|V| = 100$ was about 30 min. The algorithm VNS^2, which uses the Kernighan–Lin neighborhood on the local search phase, showed the worst results in terms of accuracy. It is interesting to note that the VNS^3, which uses Kernighan–Lin neighborhood and implements the idea of the simulated annealing approach, showed significantly better results in comparison with the algorithm VNS^2. Its running time for instances of dimension $|V| = 100$ does not exceed 150 s.

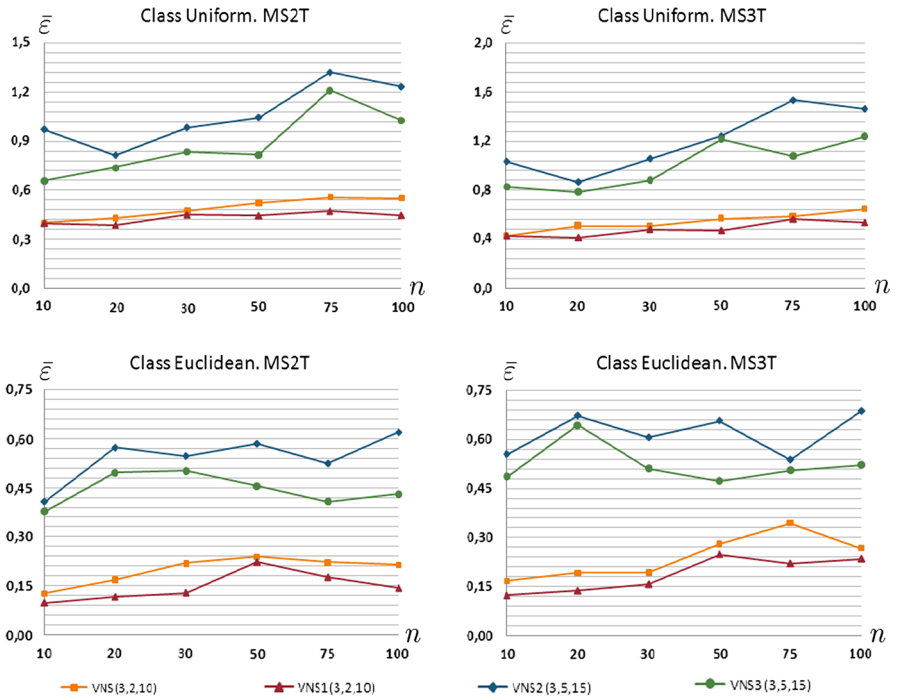

Fig. 4 The average relative error of the variable neighborhood search algorithms

6.3 Comparison of the effectiveness of the proposed algorithms

This section presents the results of the experiments to analyze the proposed algorithms in comparison with known metaheuristics: TABU SEARCH (Beltran and Skorin-Kapov 1993), SA (Simulated Annealing) (Candia and Bravo 2002) and GA (Genetic Algorithm) (Ghashghai and Rardin 2002). The efficiency was evaluated on random and real-word data sets.

Tables 2 and 3 present results of the experiments to analyze the effectiveness of the algorithms on the random data set. The experiments show that the algorithm VNS[1] demonstrated the smallest error, which on the Uniform class was less than 0.25 %. Note that this algorithm significantly outperformed all known metaheuristics in terms of accuracy. The algorithms GREEDY[3], SA and GA showed almost the same results in terms of accuracy, surpassing each other on different instances. The GREEDY[3] is insignificantly worse than the TABU SEARCH.

Table 4 demonstrates the results on the real-world data provided by "TeleGeography" consultancy. The data includes information about 25 hub-level nodes of the European backbone IP-network and the matrix of shortest distances between the nodes.

On the real data the metaheuristic VNS[1] demonstrates the best results in terms of accuracy. The algorithm VNS[3] outperformed all known metaheuristics. Figure 5 shows the solution of the MS2T problem with the real-world data built by the algorithm VNS[1]. Note that this solution is very similar to the existing structure of the IP-network

Table 2 Comparison of the efficiency of the proposed algorithms

Algorithms	$\|V\| = 10$	$\|V\| = 20$	$\|V\| = 30$	$\|V\| = 50$	$\|V\| = 75$	$\|V\| = 100$
GREEDY^3						
$\bar{\varepsilon}$	1.75	2.09	2.23	2.43	2.42	2.45
$\bar{t}$	0.00	0.04	0.24	2.11	12.32	39.30
σ	0.63	0.87	0.79	0.83	0.91	0.77
VNS^1(3, 2, 10)						
$\bar{\varepsilon}$	**0.46**	**0.39**	**0.45**	**0.44**	**0.47**	**0.45**
$\bar{t}$	0.29	4.21	18.24	127.01	609.33	1,852.61
σ	0.11	0.07	0.10	0.08	0.12	0.11
VNS^3(3, 5, 15)						
$\bar{\varepsilon}$	1.05	1.16	1.36	1.86	1.98	2.30
$\bar{t}$	0.11	1.03	3.53	16.5	47.3	125.25
σ	0.27	0.29	0.37	0.31	0.41	0.38
TABU SEARCH						
$\bar{\varepsilon}$	1.48	1.76	1.94	1.74	2.14	2.27
$\bar{t}$	1.49	23.71	36.65	303.29	554.47	2,245.95
σ	0.29	0.45	0.36	0.21	0.38	0.42
SA						
$\bar{\varepsilon}$	1.91	1.61	2.27	2.14	2.65	2.32
$\bar{t}$	1.46	11.03	24.45	99.68	186.02	466.69
σ	0.45	0.39	0.57	0.41	0.43	0.45
GA						
$\bar{\varepsilon}$	2.25	2.01	2.72	1.97	2.15	2.26
$\bar{t}$	0.05	0.44	1.80	8.38	62.38	189.98
σ	0.47	0.38	0.49	0.59	0.71	0.51

Class Uniform. MS2T problem
Bold values indicates the best results (minimal error)

but at the same time there are some differences. The differences are largely due to unaccounted conditions in the MSkT problem, such as probability of link failure, probability of node failure, bandwidth restrictions, etc.

7 Concluding remarks

In this paper we have considered the minimal spanning k-tree problem in a complete weighted graph. We have proposed algorithms based on the greedy strategy and the variable neighborhood search metaheuristics.

We have conducted numerical experiments to analyze the effectiveness of the proposed algorithms on random and real-word data sets. The experiments showed that VNS^1 outperformed metaheuristics TABU SEARCH, SA and GA and all other algorithms proposed in this paper. The average error of this algorithm on random instances does not exceed 0.25 % and the running time for instances of dimension $|V| = 100$

Table 3 Comparison of the efficiency of the proposed algorithms

Algorithms	$\|V\| = 10$	$\|V\| = 20$	$\|V\| = 30$	$\|V\| = 50$	$\|V\| = 75$	$\|V\| = 100$
GREEDY^3						
$\bar{\varepsilon}$	0.46	0.49	0.61	0.63	0.97	0.83
$\bar{t}$	0.00	0.04	0.26	2.31	12.38	34.37
σ	0.09	0.07	0.12	0.11	0.19	0.26
VNS^1(3, 2, 10)						
$\bar{\varepsilon}$	**0.10**	**0.12**	**0.13**	**0.22**	**0.18**	**0.19**
$\bar{t}$	0.28	3.97	20.45	151.26	638.17	2,102.04
σ	0.01	0.04	0.03	0.05	0.09	0.07
VNS^3(3, 5, 15)						
$\bar{\varepsilon}$	0.31	0.24	0.39	0.49	0.55	0.59
$\bar{t}$	0.13	1.13	3.37	16.22	48.39	118.35
σ	0.08	0.09	0.04	0.03	0.12	0.15
TABU SEARCH						
$\bar{\varepsilon}$	0.33	0.28	0.39	0.51	0.53	0.53
$\bar{t}$	1.51	21.64	36.48	257.55	588.06	1978.21
σ	0.09	0.08	0.12	0.15	0.13	0.17
SA						
$\bar{\varepsilon}$	0.38	0.62	0.52	0.75	0.54	0.75
$\bar{t}$	1.39	10.43	28.23	86.65	187.96	449.36
σ	0.10	0.09	0.19	0.25	0.20	0.37
GA						
$\bar{\varepsilon}$	0.32	0.45	0.57	0.85	0.89	0.95
$\bar{t}$	0.05	0.43	1.79	7.47	27.27	95.85
σ	0.06	0.11	0.18	0.23	0.22	0.33

Class Euclidean. MS2T Problem
Bold values indicates the best results (minimal error)

Table 4 Comparison of the proposed algorithms on the real-world data

Algorithms	t	ε
GREEDY^3	0.19	0.42
VNS^1(3, 2, 10)	15.82	**0.15**
VNS^3(3, 5, 15)	3.01	0.21
TABU SEARCH	32.51	0.34
SA	19.61	0.56
GA	1.49	0.63

Bold values indicates the best results (minimal error)

does not exceed 35 min. The algorithm VNS^3, which uses Kernighan–Lin neighborhood on the local search phase and partially implements the idea of the simulated annealing approach showed almost the same accuracy with the known metaheuristics. The running time of the algorithm VNS^3 for solving the problem of dimension $|V| = 100$ does not exceed 125 s. The results indicate prospects of use the algorithm

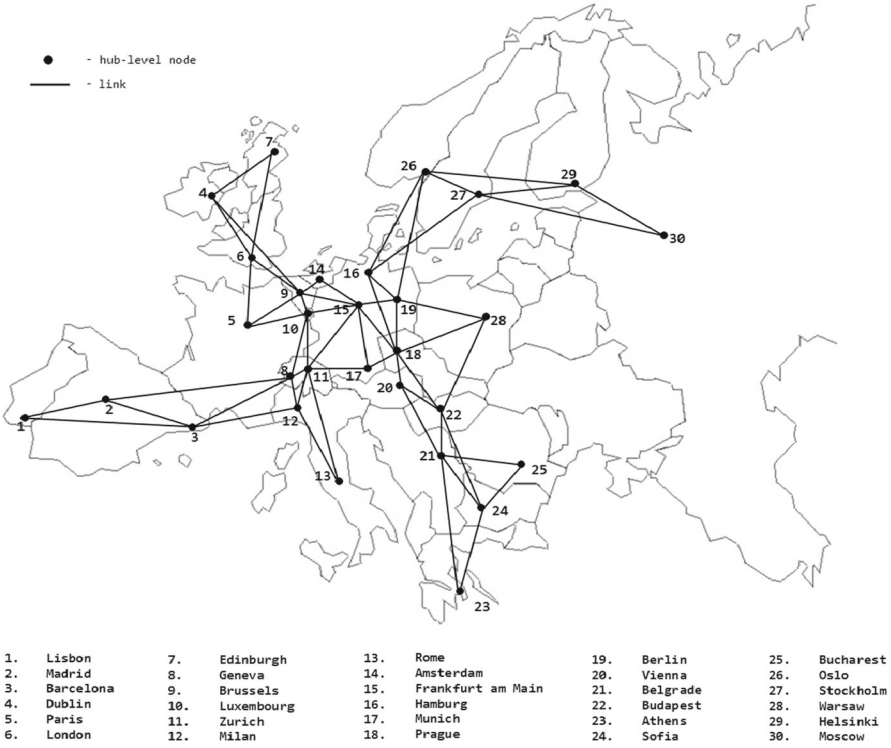

Fig. 5 The solution of the MS2T with the real-world data built by the algorithm VNS[1]

for solving problems of large and extra large dimensions, while the algorithm VNS[1], due to its high time complexity, is advised to be applied for solving problems of small and medium dimensions.

Importantly, the numerical experiments on the real-world data showed that the MSkT problem in its pure form is interesting from more theoretical than practical point of view because its mathematical model does not take into account many conditions that are critical in designing real-world communication networks: probability of link failure, probability of node failure, bandwidth restrictions, etc. Based on this, as a direction of the further research, it seems appropriate to study modifications of the MSkT problem which take into account conditions listed above. This direction seems to be promising for the following reasons. First, many real-world communication networks are partial 3-trees (i.e. connected spanning subgraphs of a 3-tree) (Granot and Skorin-Kapov 1994). Second, reliability of k-trees, due to their connectivity properties, outperforms reliability of many well-known topologies, for instance, tree topology, ring topology, etc (Candia and Bravo 2002). Third, the problem of calculating a network reliability measure, which is NP-complete on arbitrary graphs, is solvable in polynomial time on k-trees (for fixed k) (Granot and Skorin-Kapov 1994).

Acknowledgments The authors are grateful to professors D. Skoryn-Kapov and A. Koster for their interest in the problem and constructive suggestions, to professors F. Beltran, A. Candia and G. Fernandez for providing the codes of known metaheuristics. Research by P. Pardalos was conducted at National Research University Higher School of Economics and supported by RSF Grant 14-41-00039.

References

Alvarez-Mirandaa E, Ljubicb I, Toth P (2013) Exact approaches for solving robust prize-collecting Steiner tree problems. Eur J Oper Res 229:599–612

Bansal N, Khandekar R, Konemann J (2013) On generalizations of network design problems with degree bounds. Math Program 141:479–506

Beck H, Candia A (1993) An heuristic for the minimum spanning 2-tree problem. Comput Sci 47:97–109

Beck H, Candia A, Bravo H (1993) Optimal design of invulnerable networks. Res Report 15:107–113

Beck H, Candia A (2000) Heuristics for minimum spanning k-trees. Invest Oper 9:104–116

Beltran H, Skorin-Kapov D (1993) On minimum cost isolated failure immune network. Telecommun Syst Model Anal 12:444–453

Bern M (1987) Networks design problems: Steiner trees and spanning k-trees, Ph. D. Thesis. University of Berkeley

Cai L, Maffray F (1992) On the spanning k-tree problem. University of Toronto, Toronto

Cai L (1996) On spanning 2-trees in a graph. University of Hong Kong, Hong Kong

Candia A, Bravo HA (2002) Simulated annealing approach for minimum cost isolated failure immune networks. Ess. Surv. Metaheuristics. 15:169–183

Dengiza B, Altiparmak F, Belgin O (2010) Design of reliable communication networks: A hybrid ant colony optimization algorithm. IIE Trans 42:891

Dinh T et al (2012) On new approaches of assessing network vulnerability: hardness and approximation. IEEE/ACM Trans Netw 20:609–619

Elshqeirat B, Soh S, Lazarescu M (2013) Dynamic programming for minimal cost topology with reliability constraint. Adv Comput Netw 1:286–290

Farley A (1981) Networks immune to isolated failures. Networks 11:255–268

Garey M, Johnson D (1976) Some simplified NP-complete graph problems. Theor Comput Sci 1:237–267

Ghashghai E, Rardin R (2002) Using genetic algorithms to find good k-tree subgraphs. Evol Optim 48: 399–413

Golumbic M (1980) Algorithmic graph theory and perfect graphs. Academic Press, New York

Granot D, Skorin-Kapov D (1994) On some optimization problems on k-trees and partial k-trees. Discre Appl Math 48:129–145

Grotschel M, Monma C (1992) Facets for polyhedra arising in the design of communication networks with low-connectity constraints. SIAM J Optim 2:474–504

Haidine A (2013) Design of reliable fiber-based distribution networks modeled by multi-objective combinatorial optimization. Int J Commun Syst 26:1227–1242

Hansen P, Mladenovic N, Brimberg J (2010) Handbook of Metaheuristics. Intern Series Oper Res Manage Sci 146:61–86

Johnston M, Lee H, Modiano E (2013) Robust network design for stochastic traffic demands. J Lightwave Technol 31:3104–3116

Khandekara R, Kortsarzb G, Nutovc Z (2013) On some network design problems with degree constraints. J Comput Syst Sci 79:725–736

Konak A, Smith A (2011) Efficient optimization of reliable two-node connected networks: a biobjective approach. Inf Sci Technol 23:430–445

Magnanti TL, Raghavan S (2005) Strong formulations for network design problems with connectivity requirements. Networks 45:61–79

Pardalos P, Thai MT (2011) Handbook of optimization in complex. Springer, New York

Prim R (1957) Shortest connection networks and some generalizations. Bell Syst Technol J 36:1389–1401

Rose D (1970) Triangulated graphs and the elimination process. J Math Anal Appl 32:597–609

Rose D (1974) On simple characterizations of k-trees. Discret. Math. 41:317–322

Shi Q (2008) Efficient algorithm for network center/covering location optimazation problems. Ph.D. thesis, School of Computing Science-Simon Fraser University

Simonetti L, Protti F, Frota Y (2011) New branch-and-bound algorithms for k-cardinality tree problems. Electron Notes Discret Math 37:27–32

Thai MT, Pardalos P (2011) Handbook of optimization in complex networks: theory and applications. Springer, New York

Wald J, Colbourn C (1983) Steiner trees, partial 2-trees and minimum IFI networks. Networks 13:159–167