Soft Computing
https://doi.org/10.1007/s00500-018-3557-3

FOCUS

The Minimum Routing Cost Tree Problem

State of the art and a core-node based heuristic algorithm

Adriano Masone[1] · Maria Elena Nenni[2] · Antonio Sforza[1] · Claudio Sterle[1,3]

Abstract
The minimum routing cost tree problem arises when we need to find the tree minimizing the minimum travel/communication cost, i.e., the tree which presents the minimal difference with the same cost computed on the whole network. This paper provides the state of the art of the problem and proposes a new heuristic based on the identification of a core of the network around which the solution can be built. The algorithm has been tested on literature instances of up to one thousand nodes. The results, compared with those of other heuristic algorithms, prove the competitiveness of the proposed one both in terms of the quality of the solution and computation time.

Keywords Network design · Spanning tree · Routing cost · Shortest path

1 Introduction

Network design problems arise in many application fields: traffic and transportation, water resource management and communication, to cite only a few. For these design problems, given an undirected network where the nodes represent plants or activities, one or more paths are available for each pair (u, v) of nodes and it is possible to define a minimum cost $c(u, v)$ which could be a distance or a travel/communication cost. The total travel/communication cost is the cost obtained by summing the costs over all the pairs (u, v) of nodes. Sometimes it is necessary to design a spanning tree of the given network. In these cases, on any spanning tree T of the network there is only one path for each pair (u, v) and we need to find the tree which has the minimum travel/communication cost, i.e., the tree which presents the minimal difference with the same cost computed on the whole network. This problem, known as the minimum routing cost tree (MRCT) problem, has received attention in some application fields. For example, it arises to solve the problem of the alignment of genomic sequences in biology (Fischetti et al. 2002). Furthermore, it is particularly present in peer-to-peer telecommunication networks where an efficient tree has to be designed (Campos and Ricardo 2008). This work focuses on the little dealt with minimum routing cost tree problem, providing a review of the main contributions present in literature, classified in exact, heuristic and metaheuristic approaches. Moreover, the paper presents a new heuristic algorithm to tackle it, based on the identification of a core of the network around which the solution is built. The algorithm has been tested on literature instances of up to one thousand nodes. The results, compared to the ones of other heuristic algorithms, show the competitiveness of the proposed heuristic both in terms of objective function value and computation times.

The paper is organized as follows: Sect. 2 presents two mathematical formulations of the problem, Sect. 3 contains an overview of the main contributions proposed in the liter-

Communicated by P. Beraldi, M. Boccia, C. Sterle.

✉ Adriano Masone
adriano.masone@unina.it

Maria Elena Nenni
mariaelena.nenni@unina.it

Antonio Sforza
sforza@unina.it

Claudio Sterle
claudio.sterle@unina.it

[1] Department of Electrical Engineering and Information Technology, University "Federico II" of Naples, Via Claudio 21, Naples, Italy

[2] Department of Industrial Engineering, University "Federico II" of Naples, Via Claudio 21, Naples, Italy

[3] Istituto di Analisi dei Sistemi ed Informatica "A. Ruberti", CNR, Rome, Italy

Published online: 10 October 2018

ature, Sect. 4 describes a new heuristic, Sect. 5 compares the computational results of the proposed heuristic with those of other algorithms on work bench instances and, finally, Sect. 6 presents some conclusions and future work perspectives.

2 Problem formulation

Given an undirected network with nonnegative edge weights $N(V, E, W)$, where V is the set of nodes ($|V| = v$), E is the set of edges ($|E| = m$) and W is the set of edge weights, we will indicate with w_e, $w_e \geq 0$, $\forall e \in E$, the weight of the edge e. Let r be a generic pair of nodes (u, v) and let R be the set of all pairs of nodes. Let $c(r)$ be the shortest path cost between the two nodes. As previously stated, on a tree T there is only one path for each pair of nodes with a cost $c(r, T)$. The routing cost of a spanning tree T is defined as $C(T) = \sum_{r \in R} c(r, T)$. The MRCT problem consists of finding the tree minimizing the routing cost summed over all pairs of nodes. The MRCT problem is proved to be NP-hard in Johnson et al. (1978). Two linear mixed-integer programming (MIP) formulations have been presented in Fischetti et al. (2002). The first is a path-based formulation while the second is a flow-based formulation, both reported in the following subsections.

2.1 Path-based formulation

Given a pair of nodes $r, r \in R$, we denote by P_r the set of all (simple) paths in N between u and v. Let P be the set of all paths for all pairs. The cost for each path p, $p \in P$, will be $c_p = \sum_{e \in p} w_e$. Let $x_p, \forall p \in P$, be the decision variable associated to a path p for each pair of nodes. Let $y_e, \forall e \in E$, be the variable associated to an edge e of a tree solution. On this basis, the path-based MIP model is the following:

$$\min C_1 = \sum_{p \in P} c_p x_p \tag{1}$$

$$\sum_{p \in P_r} x_p = 1 \qquad \forall r \in R \tag{2}$$

$$\sum_{p \in P_r : p \ni e} x_p \leq y_e \qquad \forall e \in E, \forall r \in R \tag{3}$$

$$\sum_{e \in E} y_e = n - 1 \tag{4}$$

$$y_e \in \{0, 1\} \qquad \forall e \in E \tag{5}$$

$$x_p \geq 0, \qquad \forall p \in P \tag{6}$$

The objective function (1) minimizes the routing cost summed over all the paths. The constraints (2) guarantee that each pair of nodes is connected by a path. Constraints (3) ensure that if a pair r is connected by a path p, the related edges will be selected. The constraint (4) forces the number of edges to be used to $n - 1$. The constraints (5) and (6) are related to the nature of the variables.

2.2 Flow-based formulation

The MRCT problem can also be formulated as a multicommodity flow problem, where each commodity corresponds to a pair r of nodes, supposing that each node sends one flow unit to each other $n - 1$ nodes. Let A be the set of directed arcs obtained by adopting two directed arcs for each edge of E in both directions. Therefore, two arcs ij and ji belong to A for each $e = (ij) \in E$ and $w_{ij} = w_{ji} = w_e$. Let x_{ij}^r be the real variable associated to the directed arc ij and commodity r, equal to the flow which traverses the arc ij from u to v. Moreover, let $y_e, \forall e \in E$, be a binary variable which is equal to 1 if the edge e belongs to a tree, 0 otherwise. With this notation and setting, the flow-based formulation is the following:

$$\min C_2 = \sum_{r \in R} \sum_{ij \in A} w_{ij} x_{ij}^r \tag{7}$$

$$\sum_{j:uj \in A} x_{uj}^r - \sum_{k:ku \in A} x_{ku}^r = 1 \qquad \forall r = \{u, v\} \in R, \forall u \in V, u < v \tag{8}$$

$$\sum_{j:ij \in A} x_{ij}^r - \sum_{k:ki \in A} x_{ki}^r = 0 \qquad \forall r = \{u, v\} \in R, \forall i \in V - \{u, v\} \tag{9}$$

$$x_{ij}^r + x_{ji}^r \leq y_e, \qquad \forall e = (ij) \in E, \forall r \in R \tag{10}$$

$$\sum_{e \in E} y_e = n - 1 \tag{11}$$

$$y_e \in \{0, 1\}, \qquad \forall e \in E \tag{12}$$

$$x_{ij}^r \geq 0, \qquad \forall r = \{u, v\} \in R, ij \in A \tag{13}$$

The objective function (7) minimizes the total routing cost computed over all the arcs and all the commodities. The continuity constraints (8) in the node u for the commodity r force the variables x_{ij}^r to be 0 or 1. Constraints (9) guarantee the balance of the flow in the transit nodes for the pair r. Constraints (10) allow to send the flow related to the commodity r only on an edge of the tree. The constraint (11) forces the number of edges to be used to $n - 1$. The constraints (12) and (13) are related to the nature of the variables.

3 Literature review

According to the problem solving method used, papers on MRCT can be classified in the following categories:

– an exact approach
– a heuristic approach
– a metaheuristic approach

In this work we will focus on the MRCT problem in the basic form described above. For the sake of completeness, the reader interested in other variants of the problem (the two-source minimum routing cost spanning tree, the degree constrained minimum routing cost tree, the minimum routing cost clustered tree problem) is referred to specialized papers (Wu 2002; Chen et al. 2007; Kim et al. 2015; Lin and Wu 2017).

3.1 The exact approach

To the best of the authors knowledge, the only exact algorithm developed to tackle the MRCT problem is presented in Fischetti et al. (2002), where a Branch and Bound approach specialized for the path-based formulation is proposed. Due to the huge number of variables, the approach is characterized by a column generation procedure. The authors also studied the two formulations described above in depth, comparing them in terms of number of variables, constraints and quality of the corresponding linear relaxation. Moreover, to improve the performance of the Branch and Bound approach the authors proposed some valid inequalities and a general-purpose strategy, defined "LP-Shortcut", to efficiently generate rows and columns. The approach has been tested on euclidean and random networks of up to fifty nodes (Table 1). Despite the increase of computational resources, the theme of exact approaches has scarcely been investigated and their usage is limited to small instances. This is mainly motivated by the need arising in real applications to solve the problem with very low computation times. This led the research activity to focus on the heuristic and metaheuristic methods surveyed in the following subsections.

3.2 The heuristic approach

Heuristic approaches have received great deal of attention in MRCT literature, as in many applications a good solution has to be determined with very low computation times. Heuristic approaches are needed in these cases due to the complexity of the problem. The first, and probably the best known heuristic algorithm for the MRCT problem is proposed by Wong (1978) in Wong (1980). The algorithm is based on the computation of the shortest path tree from each node, choosing the shortest path tree with the minimum routing cost as a solution. The complexity of the algorithm is $O(nm + n^2 \log n)$. The author does not provide computational tests, but he proves that it is a 2-approximation algorithm.

In Wu et al. (1999) the authors propose a scheme for finding in polynomial time ($O(n^{2\lfloor 2/\epsilon \rfloor - 2})$ a k-star of the network, that is a tree with at most k internal (not leaves) nodes, where k is a function of ϵ ($k = 2/\epsilon - 1$). They prove that the k-star with the lowest routing cost is a $(1+\epsilon)$-approximate solution of the MRCT problem.

Campos and Ricardo (2008) propose a heuristic method to compute a MRCT solution in Campos and Ricardo (2008). The algorithm is an adaptation of Prim's shortest path algorithm (Prim 1957). Unlike Prim's algorithm, the initial node is chosen on the basis of the degree of the node, the sum of adjacent edge weights and the maximum adjacent edge weight. Then, at each iteration, an edge is added to the current sub-tree considering a combination of several parameters (e.g., weight of the edge, estimated cost of the path between the node destination of the edge and all the nodes which belong to the partial solution) instead of the edge weight. The algorithm has been tested on instances of up to fifty nodes. This algorithm provides solutions with a value comparable with those of Wong's algorithm with computational times generally twice those of Prim's algorithm.

The algorithm proposed in Singh (2008) determines an initial solution using another variant of Prim's algorithm (Prim 1957). The starting node is randomly chosen and the edges to be added are selected using a probabilistic parameter built on the basis of the reciprocal of the weight. The solution obtained in this way is refined through a local search approach. It randomly deletes an edge of the solution and tries to connect the two resulting components by adding the other edges one by one. The edge that results in the tree with the minimum routing cost is added. If no improving solution is found for a prefixed number of iterations, the algorithm tries to improve the current solution by a random perturbation of the current solution. The algorithm is tested using the instances proposed in Julstrom (2005) of up to three hundred nodes with computational times up to 600 s.

Three heuristics based on the removal of the edges are proposed in Tan (2012a).The first algorithm starts from the initial network and then eliminates an edge at each iteration, so that the sum of the routing costs on the graph is minimum, until a tree is obtained. The second algorithm computes an initial tree using Prim's algorithm and then it applies the cycle local search defined in Wolf and Merz (2010) to improve the solution. In the third heuristic, called the bridge-processing algorithm, at each iteration the current network is randomly split into two sub-networks deleting all the edges lying between the two sub-networks. An edge is then added so that the resulting network has the minimum routing cost. The three heuristics are compared in terms of objective function value and computational time, testing them on randomly generated instances of up to five hundred nodes. The first heuristic is not able to tackle the bigger instance within 1 h while the computation times of the second and third heuristics on the bigger instance are about 200 s.

Table 1 summarizes the maximum size of the tackled networks and the related computational times, as reported in the original paper.

3.3 The metaheuristic approach

The first metaheuristic contribution for the MRCT problem is proposed in Julstrom (2005). In this work the author presents a particular encoding based on the usage of the Prufer strings and the Blob Code (Julstrom 2001) to be used in genetic/evolutionary algorithms. The Prufer strings have a length of $n - 2$ over an alphabet of n symbols. The Blob Code is a one-to-one mapping between Prufer strings and spanning trees. The computational results on networks of up to three hundred nodes show that this encoding is competitive with the classical edge-set one (Raidl and Julstrom 2003) with computation times on the bigger instances up to 300 s.

In Merz and Wolf (2006) three local search methods, based on the repositioning of sub-trees in a tree, are presented to be used within an evolutionary algorithm. In Wolf and Merz (2010) the same authors present their evolutionary algorithm improved by a cycle local search, where in each iteration an edge is added to the tree, thus generating a cycle. The tree topology is restored by removing the edge whose deletion results in the tree of minimum routing cost. The proposed approaches are evaluated by testing them on instances of up to four hundred nodes, originating from a real application in the telecommunication field with computation times on the bigger instances up to 200 s.

A genetic algorithm is proposed in Tan (2012b). The authors show that among different kinds of encodings (binary, edge-based, node-based, etc.), which can be used to represent a tree, the edge-based one is the most suitable to be used in a genetic algorithm. The proposed algorithm is tested on randomly generated instances of up to two hundred nodes.

An artificial bee colony algorithm is described in Singh and Sundar (2011), where a solution is represented by a bee which is assigned to a source food whose amount of nectar corresponds to the related objective function value. A local search is applied at the end of the algorithm. It consists of the deletion of one edge at a time and the connection of the resulting two sub-trees, made by adding the edge whose inclusion results in the tree of minimum routing cost. The algorithm is tested using the instances proposed in Julstrom (2005) of up to three hundred nodes with computation times on the bigger instances up to 300 s.

An ant colony algorithm is proposed in Hieu et al. (2011) where an edge corresponds to a trail that the ants could follow characterized by a certain level of pheromone proportional to the weight of the corresponding edge. The ants gradually select edges and add them to the solution until it is a spanning tree of n nodes. The algorithm is tested on randomly generated instances with up to one hundred nodes.

A variable neighborhood search (VNS) specialized for the MRCT problem is presented in Sattari and Didehvar (2013). The search starts from an initial solution computed using Wong's algorithm (Wong 1980) and then, at each iteration, enhances it considering a suitable k-neighborhood, defined by all the trees that differ from the current tree in k edges, for $k \in [1, n - 2]$. The algorithm is tested using the instances proposed in Julstrom (2005) of up to three hundred nodes with computation times on the bigger instances up to 300 s.

The same authors proposed a greedy randomized adaptive search procedure in Sattari and Didehvar (2015). Each solution is computed by a heuristic based on the combination of Dijkstra's and Prim's shortest path algorithms (Dijkstra 1959; Prim 1957). The solutions are improved by a local search method. Finally, a path-relinking memory-based method stores the best solutions and combines parts of the stored solutions in order to find new ones. The algorithm is tested using the instances proposed in Julstrom (2005) of up to three hundred nodes with computation times on the bigger instances up to 200 s.

A wide experimental study of several metaheuristics and heuristics is reported in Tan and Due (2013). Their performances are tested on many kinds of networks (euclidean, randomized, complete and sparse) with up to one thousand nodes.

Table 1 summarizes the maximum size of the tackled networks and the related computational times, as reported in the original paper.

4 A new core-node-based heuristic for the MRCT problem

The following subsections describe a new heuristic algorithm, based on the identification of a "core" of the network, i.e., a subset of nodes around which the solution can be built. To this end, we classify the nodes in several levels, using opportunely weighted outdegree values. The external levels represent the outlying nodes and the internal levels represent the "core" nodes, i.e., the central nodes of the MRCT. Once the nodes have been assigned to the different levels, a network reduction is operated to build the "core" of the network. A successive phase builds a tree starting from the "core". The algorithm adds the nodes belonging to the successive levels to the current core at each iteration. The tree solution thus determined is finally optimized by a local search procedure. The main phases of the algorithm and the related procedures are described as follows: Phase 1-network reduction for the "core" generation; Phase 2-tree building; Phase 3-solution improvement by a local search procedure.

Table 1 Maximum size of the instances tackled and related computational times of the algorithms

Authors	Year	$\|V\|$	T (s)
Exact			
Fischetti et al.	2002	50	> 3600
Heuristic			
Wong	1980	–	–
Wu et al.	1999	–	–
Campos and Ricardo	2008	50	Negligible
Singh	2008	300	600
Tan	2012	500	> 3600; 200; 200
Metaheuristic			
Julstrom	2005	300	300
Merz and Wolf	2006	400	200; 200; 6400
Wolf and Merz	2010	400	300
Hieu	2011	100	–
Singh and Sundar	2011	300	300
Tan	2012	200	–
Satteri and Didehvar	2013	300	300
Tan and Due	2013	1000	0–2400
Sattari and Didehvar	2015	300	200

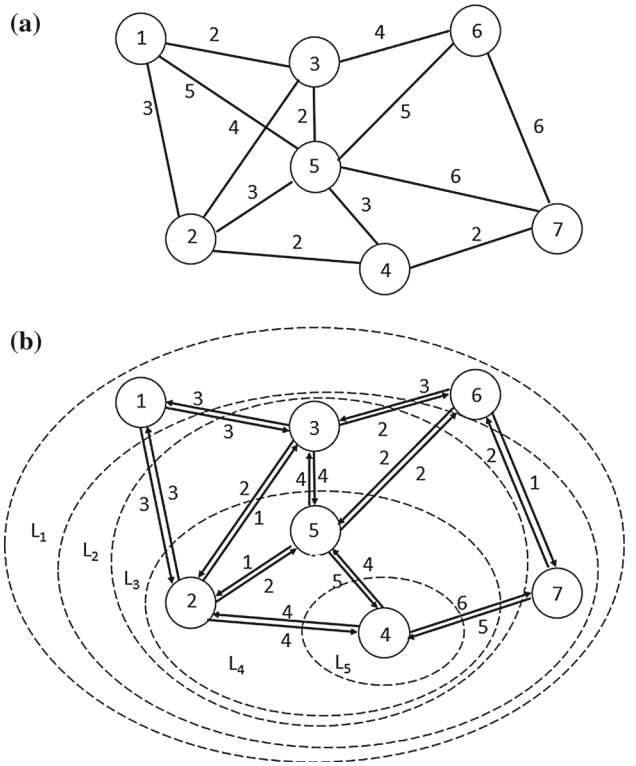

Fig. 1 **a** The network N. **b** The network N' with n_{ij}. Node classification and reduction process to core node 4

4.1 Phase 1: network reduction for the "core" generation

We compute the shortest path tree for each node in N using one of the well-known algorithms.

Then, on the basis of these shortest path trees, we build a directed network $N'(V, A')$ where V is the set of nodes of network N and A' is the set of directed arcs, so that two arcs ij and ji exist for each edge $ij \in E$ in N.

We can define and compute two parameters, n_{ij} and n_{ji}, as the number of the origin/destination pairs o/d that use the edge ij in all the shortest path trees, going from i to j or from j to i. These parameters constitute the set of new arc weights of N', where $w'_{ij} = n_{ij}$.

If $n_{ij} = 0$, then the arc ij does not exist. If n_{ij} has a low value, the corresponding arc is used by few o/d pairs. If n_{ij} has a high value, the corresponding arc is in many o/d pair paths.

Each node in N' has an indegree (outdegree), i.e., the number of arcs entering (exiting) a node. These degrees can be weighted with the parameter n_{ij}, thus generating the weighted indegree $d_{k,\text{in}}$ and outdegree $d_{k,\text{out}}$ for a node k:

$$d_{k,\text{in}} = \sum_{i:ik \in A'} n_{ik}, \quad d_{k,\text{out}} = \sum_{j:kj \in A'} n_{kj}.$$

The nodes with a low weighted outdegree can be assumed as candidates to be leaves of the MRCT. The nodes with a high weighted outdegree can be assumed as candidates to be core nodes of the MRCT. This allows us to classify the nodes in successive levels with an increasing weighted outdegree.

On the basis of this classification the algorithm removes from N' all the nodes with the lowest $d_{k,\text{out}}$ and all the arcs entering and exiting them, thus generating a reduced network N''.

The same reduction is performed on N'' thus generating a new reduced network, until a core node or a set of core nodes with the same weighted outdegree is determined.

The reduction procedure is a fundamental phase of the algorithm, since the identification of the core of the network will strongly affect the structure of the final solution. The pseudocode of the network reduction to find the core node is given in Algorithm 1.

The results of this phase are the level to which a node k has been assigned ($NodeLevel(k)$) and the set of nodes assigned to each level l, indicated as ($L(l)$). The level of the core will be indicated as $l_{\max}$. Figure 1a reports a simple test network N of 7 nodes and 13 edges, Fig. 1b shows the directed network N', where the arc weights are the parameters n_{ij} that express the number of pairs that use each arc. The same Fig. 1b shows the node classification and the reduction process from the whole network to the core node 4. This process is summarized in Table 2, where the weighted outdegree $d_{k,\text{out}}$ computed at each iteration of the reduction is reported.

Table 2 Weighted outdegree $d_{k,\text{out}}$ at each iteration of the reduction process

	$d_{k,\text{out}}$ Iterations				
k	1	2	3	4	5
1	6	–	–	–	–
2	10	7	7	6	–
3	11	6	6	–	–
4	14	14	8	8	0
5	12	10	10	6	–
6	6	–	–	–	–
7	7	5	–	–	–

Algorithm 1 Pseudocode - Network reduction to find the core node

Initialize $V' := V$.
for all $k \in V$ **do**
  Compute the shortest path tree $T(k)$
  for all $ij \in T(k)$ **do**
    Compute $n_{ij}^{T(k)}$
  end for
end for
for all $ij \in A$ **do**
  $n_{ij} := \sum_{k \in V'} n_{ij}^{T(k)}$
end for
repeat
  $l++$
  for all $k \in V'$ **do**
    $d_{k,out} := \sum_{j \in V'} n_{kj}$
  end for
  $d_{min} := \min_{k \in V'} d_{k,out}$
  $L(l) := \{k \in V' | d_{k,out} = d_{min}\}$
  $V' - = L(l)$
  for all $k \in L(l)$ **do**
    $NodeLevel(k) := l$
  end for
until $V' = \emptyset$
$l_{max} := l$
return L, $NodeLevel$ and l_{max}

4.2 Phase 2: tree building

The tree building starts from the core node. We add nodes to it using the node classification defined above. We select the node j of the level adjacent to the core that has the maximum weighted indegree, and we connect it to the core, using an edge ij corresponding to one of several selection criteria: minimum w_{ij}, maximum n_{ij}, minimum w_{ij}/n_{ij}.

We repeat this procedure for all the nodes of the same level. In this way the core is enlarged with the nodes of this level. The procedure continues, adding the nodes of the successive levels, adjacent to the current enlarged core, following the same selected rule above. The procedure is repeated until the nodes of all the levels have been added to the initial core thus generating a tree.

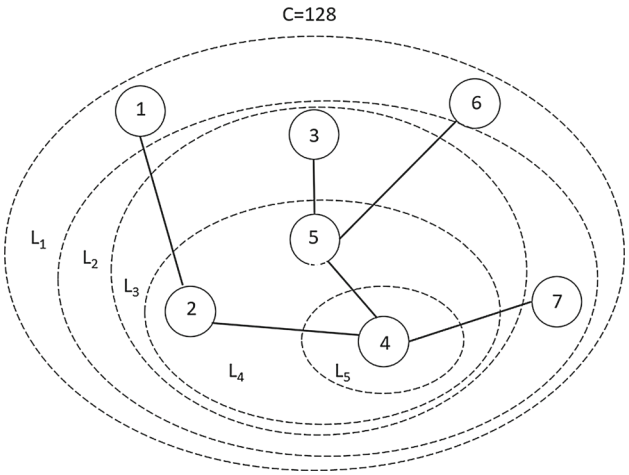

Fig. 2 Tree built starting from the core node 4

If the initial core is made by more than one node, we select the one with the maximum weighted indegree, assuming it as initial core and performing the procedure defined above.

In short, we can say that, following the ranking of the node classification, we add all the other nodes to the current core. These nodes will be connected to the current core with the edge determined by the edge selection rule. The order according to which nodes will be added to the current core depends on the node classification defined above. Indeed, we first add to the current core all the nodes of the first level, and then all the nodes of the second level and so on.

The pseudocode of the tree building (selecting the edge with the maximum n_{ij}) is given in Algorithm 2. Figure 2 shows the tree built starting from the core node 4.

4.3 Phase 3: solution improvement by a local search procedure

A local search procedure has been developed to improve the solution obtained in Phase 2. In order to explain it we have to define the following settings. Let $T(V, E_T)$ be the tree solution obtained in Phase 2 and let $S \subset V$ be the core of T, which is a node or a set of nodes. As k is a node of T there is only one chain connecting k to the core S. Therefore, we can remove the edge $kf, kf \in E_T$, incident in k and belonging to this chain to split the tree T into two sub-trees, $T_1(V_1, E_{T_1})$ and $T_2(V_2, E_{T_2})$, where V_1 and V_2 are two node subsets constituting a partition of V such that $V_1 \ni k$ and $V_2 \supseteq S$. Recall that $L(l)$ indicates the set of nodes belonging to the generic level l, sorted by decreasing value of the weighted indegree, and that $l_{\max}$ is the core level. Starting from the first node in $L(l_{\max})$, indicated as $g, g \in V$, we perform the splitting of T related to g, as described above, and we consider all the edges connecting g to the nodes of V_2 able to restore the connection of the tree. Each edge generates a

Algorithm 2 Pseudocode - Tree building with maximum n_{ij} as edge selection rule (L and l_{max} are the output of Algorithm 1)

```
Initialize T = ∅, E_T = ∅, l_1 := l_max.
repeat
    Maximum:=0
    l_2 := l_1
    repeat
        for all k ∈ L(l_1) | k ∉ T do
            CurrentNode := 0
            if T = ∅ then
                T+ = {k}
            else
                for all j ∈ T | n_jk > 0 do
                    if Maximum < n_jk then
                        CurrentNode := k
                        EdgeToAdd := jk
                        Maximum := n_jk
                    end if
                end for
                if CurrentNode ≠ 0 then
                    BREAK
                end if
            end if
        end for
        l_2 − −
    until CurrentNode ≠ 0
    T+ = {CurrentNode}
    E_T+ = {EdgeToAdd}
    if L(l_1) ⊆ S then
        l_1 − −
    end if
until T = V
return E_T
```

different tree with its related MRCT objective function value. If $T^{\wedge}$ is the best tree obtained by adding the edge gj, $j \in V_2$ to restore the connection, if $C(T^{\wedge}) < C(T)$ we update our solution substituting the removed edge gf with the edge gj. This operation is repeated for all the nodes in the level $l_{\max}$, following the above defined ranking. The procedure is repeatedly iterated from level $l_{\max} - 1$ to level 1 and then from level 1 to $l_{\max} - 1$, until no solution improvement can be achieved.

Figure 3 shows the solution obtained after the refinement of the local search which corresponds to the MRCT of the network reported in Fig. 1a.

5 Computational results

In this section we report the computational results of the proposed heuristic obtained on 21 literature instances Tan and Due (2013), which consider non-euclidean networks with edge weights belonging to the range $[1 \div 10]$. These instances have been listed for the first time in Beasley's OR-Library (Beasley 1990) for the Steiner Tree problem and can be grouped in three subsets of 7 instances each:

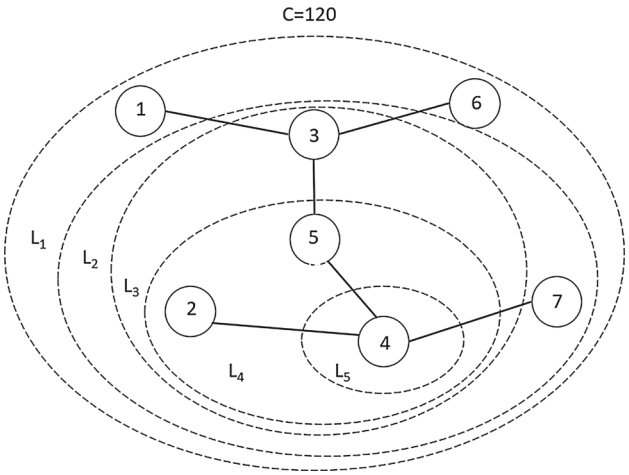

Fig. 3 Final solution after local search

- small instances, STEIB1-STEIB7, up to 75 nodes;
- medium instances, STEIC1-STEIC7, up to five hundred nodes;
- large instances, STEID1-STEID7, up to one thousand nodes;

The tests were run on an Intel Core i7-4750HQ, 2.00 GHz, 8 GB RAM, Windows 10 (64 bit) and the heuristic has been coded in C++.

The results of the different variants of the proposed algorithm, in terms both of quality of the solution and computation time, are shown in Tables 3 and 4, respectively. Table 3 shows the percentage GAPs obtained by the proposed algorithm. The instance details (name, cardinality of V and cardinality of E) are listed in the first three columns of Table 3. The Best known Upper Bound (BUB) is reported for each instance in the fourth column Tan and Due (2013). The successive columns of Table 3 show the percentage GAPs, compared to the BUB, obtained by the proposed core-based algorithm (CH). In particular, in the fifth, sixth and seventh column, the percentage GAPs of the three variants of the proposed algorithm at the end of the Phase 2 are reported. The last three columns show the percentage GAPs of the same variants at the end of Phase 3. CH-w, CH-n and CH-(w/n) indicate the columns where we report the percentage GAPs of the proposed heuristic using as an edge selection rule the criteria: minimum w_{ij}, maximum n_{ij} and minimum w_{ij}/n_{ij}, respectively. With C the objective function value of a solution obtained by an algorithm for the MRCT problem, the percentage GAP is computed as $(C - \text{BUB})/\text{BUB} * 100$. The values marked with an asterisk indicate the objective function value of the optimal solution.

From Table 3 we observe that at the end of Phase 2 the best results are obtained using the criteria CH-n and CH-(w/n), whose maximum and average GAPs are 33.57% and

Table 3 Percentage GAPs of the proposed method with different edge selection rules

INSTANCE	$\|V\|$	$\|E\|$	BUB	PHASE 2			PHASE 3		
				CH-w	CH-n	CH-(w/n)	CH-w	CH-n	CH-(w/n)
STEIB-01	50	63	26857*	0.48	2.55	0.24	0.00	0.00	0.00
STEIB-02	50	63	30301*	5.25	3.30	5.14	0.00	0.00	0.00
STEIB-03	50	63	24423*	0.97	1.47	0.83	0.00	0.96	0.00
STEIB-04	50	100	20603	10.16	3.89	4.95	0.00	0.00	0.00
STEIB-05	50	100	17203	6.58	3.66	4.11	0.00	0.00	0.00
STEIB-06	50	100	21888	13.67	17.22	17.22	5.52	5.52	5.52
STEIB-07	75	94	69684	2.57	1.23	2.57	0.00	0.00	0.00
STEIC-01	500	625	5175660	4.84	2.40	3.34	0.00	0.00	0.00
STEIC-02	500	625	4996138	3.95	3.18	2.79	0.00	0.00	0.00
STEIC-03	500	625	6102262	9.22	8.73	8.68	0.00	0.00	0.00
STEIC-04	500	625	5600166	19.86	18.18	18.33	0.00	0.00	0.00
STEIC-05	500	625	5693856	19.27	19.63	19.55	2.45	2.45	2.45
STEIC-06	500	1000	3639892	28.45	27.67	29.47	1.02	1.02	1.02
STEIC-07	500	1000	3489091	27.47	21.41	24.41	0.00	0.00	0.00
STEID-01	1000	1250	20950008	14.63	13.95	13.64	0.00	0.00	0.00
STEID-02	1000	1250	22161426	8.78	5.39	6.82	0.00	0.00	0.00
STEID-03	1000	1250	22979908	8.11	5.37	5.94	0.00	0.00	0.00
STEID-04	1000	1250	22279448	17.69	15.89	16.60	0.00	0.00	0.00
STEID-05	1000	1250	23599004	20.20	17.26	18.90	0.00	0.00	0.00
STEID-06	1000	2000	15014393	38.61	33.57	35.62	0.00	0.00	0.00
STEID-07	1000	2000	15652879	37.40	31.70	35.01	0.01	0.01	0.01
AVG-STEIB				5.67	4.76	5.01	0.79	0.93	0.79
AVG-STEIC				16.15	14.46	15.23	0.50	0.50	0.50
AVG-STEID				20.77	17.59	18.93	0.00	0.00	0.00
AVG-ALL				14.20	12.27	13.06	0.43	0.47	0.43

*indicates the optimal solution value

12.27%, and 35.62% and 13.06%, respectively. This behavior can be partially explained by considering that the selection rules CH-n and CH-(w/n) use the parameter n_{ij} which gives us greater information about the number of o/d pairs that traverse an arc considering all the shortest path trees, instead of the selection rule CH-w which considers just the weight of the edge. We can also note that, at the end of Phase 3, the local search provides a significant improvement of the average and maximum percentage GAPs, which become comparable for the three variants. Indeed, the maximum percentage GAPs of CH-w, CH-n and CH-(w/n) are equal to 5.52% (STEIB-06) while the average percentage GAPs are equal to 0.43%, 0.47% and 0.43%, respectively.

Table 4 reports the following information: the instance details, the sum of the computation times of Phase 1 and Phase 2 for each variant of the proposed algorithm (reported in the column PHASE 1–2), and the computation times of Phase 3 for each variant. We observe that the computation times of the different variants are comparable for each instance. Moreover, it is important to note that the computation times of Phase 3, which involves the significant improvement of the quality of the solutions, are lower than the sum of those in Phase 1 and Phase 2. On these instances CH-(w/n), with an average computation time of 2.033 seconds (obtained by summing the average computation time of the first two phases with that of Phase 3), turns out to be slightly faster than CH-w and CH-n whose average computation times are 2.081 seconds and 2.079 seconds, respectively.

The comparison of the results of CH-(w/n), that is the variant with the lowest average GAP and computation time, with those of Wong (1980) and Campos' Campos and Ricardo (2008) algorithms, taken from Tan and Due (2013), is reported in Tables 5 and 6.

Table 5 shows the percentage GAPs compared to the BUB obtained by Wong's and Campos' algorithm (reported in the columns WONG and CAMPOS, respectively) and by the variant CH-(w/n) of the proposed algorithm, where the best percentage GAP for each instance is highlighted in bold.

Table 4 Computation times of the proposed method with different edge selection rules

INSTANCE	$\|V\|$	$\|E\|$	PHASE 1–2			PHASE 3		
			CH-w	CH-n	CH-(w/n)	CH-w	CH-n	CH-(w/n)
STEIB-01	50	63	1.011	1.304	0.942	0.000	0.001	0.000
STEIB-02	50	63	0.924	0.893	0.931	0.001	0.002	0.001
STEIB-03	50	63	1.032	0.933	0.965	0.001	0.000	0.001
STEIB-04	50	100	0.928	0.908	0.953	0.001	0.001	0.001
STEIB-05	50	100	0.920	0.947	0.932	0.001	0.002	0.002
STEIB-06	50	100	0.931	0.912	0.930	0.001	0.002	0.002
STEIB-07	75	94	0.935	0.917	0.928	0.002	0.002	0.002
STEIC-01	500	625	1.212	1.223	1.208	0.141	0.097	0.101
STEIC-02	500	625	1.287	1.218	1.198	0.142	0.099	0.103
STEIC-03	500	625	1.224	1.217	1.287	0.145	0.106	0.104
STEIC-04	500	625	1.244	1.219	1.716	0.161	0.117	0.122
STEIC-05	500	625	1.222	1.244	1.250	0.133	0.102	0.100
STEIC-06	500	1000	1.191	1.306	1.252	0.212	0.152	0.148
STEIC-07	500	1000	1.221	1.271	1.181	0.211	0.159	0.158
STEID-01	1000	1250	3.018	3.027	2.927	0.731	0.740	0.734
STEID-02	1000	1250	3.042	2.909	2.998	0.702	0.704	0.701
STEID-03	1000	1250	3.033	2.958	2.925	0.745	0.801	0.762
STEID-04	1000	1250	3.162	3.056	2.856	0.729	0.831	0.751
STEID-05	1000	1250	2.891	2.988	2.910	0.731	0.717	0.712
STEID-06	1000	2000	2.978	2.812	2.828	1.131	1.250	1.079
STEID-07	1000	2000	3.182	3.347	2.904	1.182	1.152	1.083
AVG-STEIB			0.954	0.973	0.940	0.001	0.001	0.001
AVG-STEIC			1.229	1.243	1.299	0.164	0.119	0.119
AVG-STEID			3.044	3.014	2.907	0.850	0.885	0.832
AVG-ALL			1.742	1.743	1.715	0.338	0.335	0.317

We observe that on the set of small instances Wong's algorithm returns the best results with a maximum and an average percentage GAP of 2.17% (STEIB-06) and 0.68%, respectively. Instead, those of Campos' algorithm and of CH-(w/n) are 7.84% (STEIB-04) and 2.39%, and 5.52% (STEIB-06) and 0.79%, respectively. On the set of medium instances Wong returns the lowest maximum percentage GAP, 1.66% (STEIC-02), where those of Campos' algorithm and of CH-(w/n) are 13.74% (STEIC-03) and 2.45% (STEIC-04). On these instances, CH-(w/n) turns out to be the algorithm with the lowest average percentage GAP. Indeed, CH-(w/n) average percentage GAP is 0.50% while those of Wong's and Campos' algorithms are 0.62 and 3.26%, respectively. Finally, on the set of large instances the proposed algorithm returns the best results in terms both of maximum and average percentage GAP, which are equal to 0.01% (STEID-07) and 0.00%, respectively, while those of Wong's and Campos' algorithms are 1.16% (STEID-03) and 0.56%, and 13.09% (STEID-06) and 5.38%, respectively. Considering all the instances, we note that the variant CH-(w/n) returns the best results with an average percentage GAP of 0.43% and finds the BUB on 17 instances out of 21. The average percentage Gap returned by Wong's algorithm is slightly greater than that of CH-(w/n) (0.62%) while the one returned by Campos' algorithm is 3.68%.

The average computational times on small, medium and large instances of the CH-(w/n) variant of the proposed algorithm are reported in Table 6 and compared with those of Wong's and Campos' algorithms. From this table we observe that Campos' algorithm is the fastest one, solving each instance in negligible computation times. The computation times of the proposed algorithm are slightly longer than those of Wong's algorithm but when the size of the instances increases the computation times become comparable. The greater computation times of our algorithm are still justified by the quality of the solution it returns.

6 Conclusions

This paper provides a literature review for the MRCT problem, together with a new and original heuristic. The algorithm

Table 5 Percentage GAPs of the proposed method compared with those of Wong's and Campos' algorithms

INSTANCE	$\|V\|$	$\|E\|$	BUB	WONG	CAMPOS	CH-(w/n)
STEIB-01	50	63	26857*	0.16	0.16	**0.00**
STEIB-02	50	63	30301*	0.54	0.54	**0.00**
STEIB-03	50	63	24423*	0.63	0.96	**0.00**
STEIB-04	50	100	20603	0.74	7.84	**0.00**
STEIB-05	50	100	17203	0.49	4.46	**0.00**
STEIB-06	50	100	21888	2.17	**0.95**	5.52
STEIB-07	75	94	69684	0.03	1.83	**0.00**
STEIC-01	500	625	5175660	0.39	2.15	**0.00**
STEIC-02	500	625	4996138	1.66	2.05	**0.00**
STEIC-03	500	625	6102262	0.75	13.74	**0.00**
STEIC-04	500	625	5600166	0.34	0.86	**0.00**
STEIC-05	500	625	5693856	**0.39**	2.36	2.45
STEIC-06	500	1000	3639892	**0.51**	1.50	1.02
STEIC-07	500	1000	3489091	0.30	0.16	**0.00**
STEID-01	1000	1250	20950008	0.66	0.44	**0.00**
STEID-02	1000	1250	22161426	0.13	0.31	**0.00**
STEID-03	1000	1250	22979908	1.16	7.83	**0.00**
STEID-04	1000	1250	22279448	0.13	0.20	**0.00**
STEID-05	1000	1250	23599004	0.30	8.20	**0.00**
STEID-06	1000	2000	15014393	0.40	13.09	**0.00**
STEID-07	1000	2000	15652879	1.14	7.55	**0.00**
AVG-STEIB				0.68	2.39	0.79
AVG-STEIC				0.62	3.26	0.50
AVG-STEID				0.56	5.38	0.00
AVG-ALL				0.62	3.68	0.43

*indicates the optimal solution value
The best percentage gap is indicates in bold for each instance

Table 6 Average computation times of the proposed method compared with those of Wong's and Campos' algorithms

INSTANCE	BUB	WONG	CAMPOS	CH-(w/n)
AVG-STEIB	11.40	0.00	0.00	0.94
AVG-STEIC	131.50	0.70	0.00	1.42
AVG-STEID	231.30	4.40	0.00	3.74
AVG-ALL	124.73	1.70	0.00	2.03

has been tested on literature instances of up to one thousand nodes. The results, compared to the ones of Campos' and Wong's heuristic algorithms, prove the competitiveness of the proposed method both in terms of objective function value and computation times. In conclusion, we can say that this work has a twofold objective. For expert practitioners, it represents a collection of the main findings and achievements. For researchers approaching the MRCT, it provides a broad and comprehensive insight on the topic, also posing the basis for a future research activity including the evolution of the proposed heuristic in a metaheuristic framework. Indeed, the proposed algorithm can be easily extended to the case where each pair of nodes (u, v) is characterized by a demand $d(u, v)$. This problem is referred in literature as the optimum communication spanning tree (OCT) problem and represents a generalization of the MRCT problem (Hu 1974; Wu et al. 2000; Fernndez et al. 2013).

Compliance with ethical standards

Conflict of interest Adriano Masone declares that he has no conflict of interest. Antonio Sforza declares that he has no conflict of interest. Maria Elena Nenni declares that she has no conflict of interest.

Funding The research activity of the authors was partially funded by the Department of Electrical Engineering and Information Technology and by the University Federico II of Naples, within the OPT_APP for EPG project (Optimization Approaches for designing and protecting Electric Power Grid) and MOSTOLOG project (A multi-objective approach for Sustainable Logistic System, DIETI-ALTRI_DR408_2017_Ricerca di Ateneo).

Ethical approval This article does not contain any studies with human or animals performed by any of the authors.

References

Beasley JE (1990) OR-library: distributing test problems by electronic mail. J Oper Res Soc 41:1069–1072

Campos R, Ricardo M (2008) A fast algorithm for computing minimum routing cost spanning trees. Comput Netw 52:3229–3247

Chen Y H, Liao G L, Tang C Y (2007) Approximation algorithms for 2-source minimum routing cost k-tree problems. In: Computational science and its applications, ICCSA 2007. Springer, Berlin, pp 520–533

Dijkstra EW (1959) A note on two problems in connexion with graphs. Numer Math 1:269–271

Fernndez E, Luna-Mota C, Hildenbrandt A, Reinelt G, Wiesberg S (2013) A flow formulation for the optimum communication spanning tree. Electron Notes Discrete Math 41:85–92

Fischetti M, Lancia G, Serafini P (2002) Exact algorithms for minimum routing cost trees, networks. Wiley, London, pp 161–173

Hieu NM, Quoc PT, Nghia ND (2011) An approach of ant algorithm for solving minimum routing cost spanning tree problem. In: Proceedings of the second symposium on information and communication technology, SoICT11. ACM, New York, pp 5–10

Hu TC (1974) Optimum communication spanning trees. J Comput SIAM 3:188–195

Johnson DS, Lenstra JK, Rinnooy Kan AHG (1978) The complexity of the network design problem. Networks 8:279–285

Julstrom BA (2001) The blob code: a better string coding of spanning trees for evolutionary search. In: Wu AS (ed) 2001 Genetic and evolutionary computation conference workshop program, San Francisco, CA, pp 256–261

Julstrom BA (2005) The Blob code is competitive with edgesets in genetic algorithms for the minimum routing cost spanning tree problem. In: Beyer H-G et al (eds) Proceedings of the genetic and evolutionary computation conference 2005, vol 1. ACM Press, New York, pp 585–590

Kim T, Seob SC, Kim D (2015) Distributed formation of degree constrained minimum routing cost tree in wireless ad-hoc networks. J Parallel Distrib Comput 83:143–158

Lin CW, Wu BY (2017) On the minimum routing cost clustered tree problem. J Comb Optim 33(6):1106–1121

Merz P, Wolf S (2006) Evolutionary local search for designing peer-to-peer overlay topologies based on minimum routing cost spanning trees, parallel problem solving from nature-PPSN IX. Springer, Berlin, pp 272–281

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

Raidl GR, Julstrom BA (2003) Edge sets: an effective evolutionary coding of spanning trees. IEEE Trans Evol Comput 7:225–239

Sattari S, Didehvar F (2015) A metaheuristic algorithm for the minimum routing cost spanning tree problem. Iran J Oper Res 6(1):65–78

Sattari S, Didehvar F (2013) Variable neighborhood search approach for the minimum routing cost spanning tree problem. Int J Oper Res 10(4):153–160

Singh A (2008) A new heuristic for the minimum routing cost spanning tree problem. In: Proceedings of 11th international conference on information technology. IEEE Computer Society, pp. 9–13

Singh A, Sundar S (2011) An artificial bee colony algorithm for the minimum routing cost spanning tree problem. Soft Comput Fus Found Methodol Appl 15(12):2489–2499

Tan QP (2012a) A Heuristic approach for solving the minimum routing cost spanning tree problem. Int J Mach Learn Comput, IACSIT 2:406–409

Tan QP (2012b) A genetic approach for solving minimum routing cost spanning tree problem. Int J Mach Learn Comput 2(4):410–414

Tan QP, Due NN (2013) An experimental study of minimum routing cost spanning tree algorithms. In: International conference of soft computing and pattern recognition (SoCPaR). IEEE Computer Society, pp 158–165

Wolf S, Merz P (2010) Efficient cycle search for the minimum routing cost spanning tree problem. Lect Notes Comput Sci 6022:276–287

Wong R (1980) Worst-case analysis of network design problem heuristics, SIAM. J Algebr Discr Methods 1:51–63

Wu BY, Lancia G, Bafna V, Chao KM, Ravi R, Tang CY (1999) A polynomial-time approximation scheme for minimum routing cost spanning trees. SIAM J Comp 29:761–778

Wu BY, Chao KM, Tang CY (2000) Approximation algorithms for some optimum communication spanning tree problems. Discrete Appl Math 102(3):245–266

Wu BY (2002) A polynomial time approximation scheme for the two-source minimum routing cost spanning trees. J Algorithms 44(2):359–378