Computers, Materials & Continua

Tech Science Press

https://doi.org/10.32604/cmc.2026.077825

ARTICLE

Hypergraphs for Covering Trees and Approximating Steiner Trees

Miklós Molnár*

LIRMM, University of Montpellier, CNRS, Montpellier, France

*Corresponding Author: Miklós Molnár. Email: molnar@lirmm.fr

Received: 17 December 2025; Accepted: 17 April 2026; Published: 08 May 2026

ABSTRACT: This article presents a particular tree covering technique. To cover a set of nodes belonging to an unknown tree, a set of connected small Steiner trees is proposed. These Steiner trees can be represented as hyperedges, and a chain or tree of hyperedges provides the cover. The model allows the calculation of approximate (partial) spanning trees in graphs. The idea of covering a set of nodes by hyperedges can be used directly in Steiner heuristics. The NP-hard Steiner problem in graphs is one of the most studied graph-related problems. Several heuristics are known to give approximated solutions. The classical approximations of the Steiner problem apply shortest paths. We present a generalized metric closure that can be constructed from hyperedges of limited size, and new approximations based on the generalized metric closure are proposed. A connected hyperedge set (without loops) approximates a Steiner tree. The paper also presents a performance analysis of the proposed heuristics. The variation in hyperedge size is analyzed. An interesting result is that using larger hyperedges significantly improves the algorithm's efficiency.

KEYWORDS: Graphs; spanning problems; hypergraphs; steiner problem; metric closure; approximation

1 Introduction

The minimal partial spanning tree problem, also known as the Steiner problem in graphs, is one of the most studied graph-related problems. The main objective of this article is to contribute fruitful ideas to the resolution of the Steiner problem. We propose a new method to construct approximate tree of the Steiner problem. The method is presented in two steps.

1) Our first proposal consists of analyzing the cover of an arbitrary tree (more precisely, the cover of its leaves) using k-limited hyperedges. We are interested in finding a set of connected hyperedges that is "good" in terms of cost (length). If a cycle exists in the hypergraph used, at least one hyperedge can be removed without loss of connectivity or degradation of coverage quality. Therefore, the coverage can take the form of a chain or a tree of hyperedges. These structures are cycle-free. We examine the ratio between the cost of the cover (of the hyperchain) and the cost of the base tree to be covered.

2) The proposed technique can then be applied to approximate minimal Steiner trees in graphs. Remember than these Steiner trees should cover a given node set with minimal cost. The problem is NP-hard and heuristics are needed to solve it in polynomial time. Hyperedges allow the construction of a hypergraph built on the nodes to span. In the same way as in the metric closure of a Steiner problem, which usually contains 2-limited hyperedges (i.e., shortest paths), we propose a generalized metric closure. It contains k-limited hyperedges on the set of nodes. Greedy and exact algorithms can then be proposed to construct approximations of the minimal Steiner tree.

The paper is composed of the following sections. After this Introduction, the most relevant related papers are presented. To facilitate the reading of the paper, Section 3 contains the used main notation and some definitions. The tree coverage using sets of size limited hyperedges is analyzed in Section 4. An exact method for constructing the best cost cover using hyperedges of limited size and a greedy construction can also be found. As a result of this discussion, in Section 5, possible approximation algorithms for solving the Steiner problem are presented. Some conclusions close the paper.

2 Related Work

The Steiner problem in graphs is one of the most analyzed problems related to graphs [1]. The problem is NP-hard; its decision variant is NP-complete [2]. Intensive research has been conducted to obtain approximate solutions. The classical approximations of the Steiner problem apply 2-limited hyperedges (shortest paths) and give 2-approximations [3]. Well-known algorithms are Kruskal's algorithm and Takahashi-Matsuyama's algorithm. Triplets, added Steiner nodes of small Steiner trees, have been proposed to improve the approximation ratio (firstly in [4,5]). An LP-based approximation algorithm based on an iterative randomized rounding technique has been proposed in [6]. The solution is a tree whose cost is at most $ln(4) + \epsilon < 1.39$ times the cost of an optimal Steiner tree.

The article [7] provides an overview of the rich developments from the last three decades for the STP in graphs and highlights the most recent computational studies for some of its closely related variants. As it is indicated in [8], studying various variations of Steiner trees became an exciting activity recently. The paper gives a review on important developments for the Steiner problem. A combination of three concepts: implications, conflicts, and reductions are analyzed to advance exact solutions of the Steiner tree problem and various new techniques are conceived [9,10]. By integrating the new methods into a branch-and-cut framework, an exact Steiner problem solver has been developed. In some particular cases, the problem is polynomial-time solvable is it is the case on strongly chordal graphs [11]. A Compendium on the Steiner tree problem is also available online at [12].

In recent years, the dynamic version of the Steiner problem has also attracted an attention. The effect of a sequence of edge insertions and deletions on the Steiner trees has been analyzed in [13]. To maintain a Steiner tree in the updated graph, a fully dynamic algorithm has been proposed. A later version can be found in [14]. A detailed discussion of the extension of the Steiner tree problem to dynamic graphs, an exact algorithm that computes all Steiner sets of a given size, and the application on a varying Steiner set are shown in [15,16]. An interesting result is that the dynamic Steiner problem is NP-hard even for shortest paths.

Steiner problems in metric spaces (in Euclidean or graph-defined spaces) are intimately related to hypergraphs. Earlier, in his PhD dissertation, Warme analyzed the geometric Steiner tree problem that can be viewed as a minimum-cost spanning tree in a special hypergraph, in which nodes are terminals (leaves of full Steiner trees), and hyperedges are full Steiner trees [17]. In [18], a Steiner tree representation of the hyperedges is considered as the edge representation of a hypergraph. To solve the Steiner tree problem, the metric closure (the distance graph) can be used, as it is also indicated in the chapter [19]. This chapter gives an analysis and some relations between the Steiner tree problem in graphs and the Minimum Spanning Tree problem in hypergraphs. A branch-and-cut method based on the linear relaxation of an integer program of the problem has been proposed.

Paper [20] compares several relaxations of the problem based on hypergraphs as well as relaxations based on the formulation of the Steiner problem in graphs. It indicates that the union of trees linked by the hyperedges is a graph, and that the concatenation must be reduced by solving the classical Steiner problem in this graph.

3 Definitions and Notations

Let $G = (V, E)$ be a nonempty, connected and simple graph with positive costs (lengths) on the edges. The cost of an edge $e \in E$ is $c(e)$. Let $L \subseteq V$ be a subset of nodes. Let $T = (W, F)$ be a tree in G spanning L such that the nodes in L are leaves.

Definition 1 (Diameter of a tree): *The diameter D of the tree is the path with maximum cost among all the paths in the tree:* $c(D) = max_{p \in P} c(p)$*, where P is the set of paths in the tree (usual definition).*

Definition 2 (Raceme): *We call raceme a connected subgraph* R_t *"rooted" at a node t of the diameter and not belonging to the diameter.*

t is the base/attachment/articulation node of the raceme.

The longest path D_{R_t} *from a leaf of* R_t *to the base point t is the diameter of the raceme* R_t*.*

Using these terms, a tree can be decomposed into a diameter and attached racemes. Fig. 1 illustrates the decomposition. The diameters of the two highlighted racemes are indicated near the racemes.

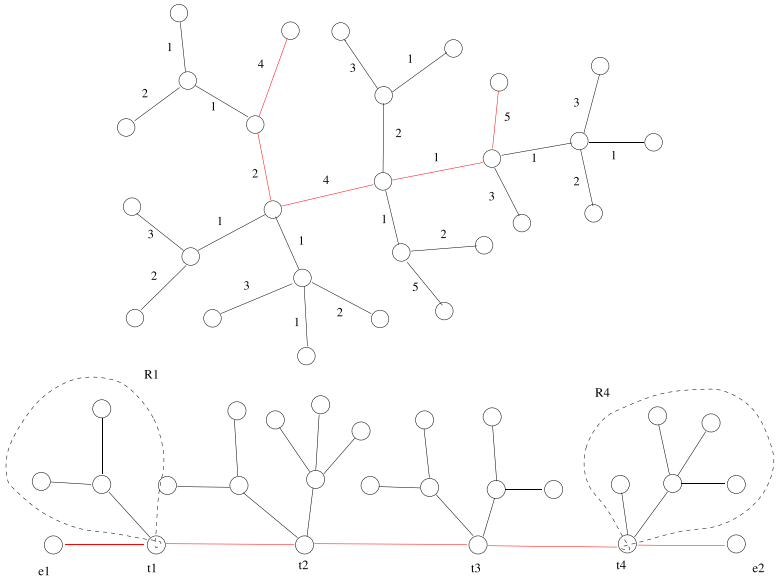

Figure 1: The decomposition of a tree into its diameter and racemes.

Property 1: *Let* e_1 *and* e_2 *be the endpoints of D. The diameter of a raceme* R_t *is not longer than the sub-paths from its base t to the extremities of the diameter D:*

$$c(D_{R_t}) \leq c(p_{t,e_1}) \quad and \quad c(D_{R_t}) \leq c(p_{t,e_2})$$

Proof: Trivial. Let us suppose that the path D_{R_t} is longer than the path p_{t,e_1}. In this case, the concatenation of paths D_{R_t} and p_{t,e_1} is longer than p_{e_1,e_2}, and this latter cannot be the diameter. □

For any raceme, an important consequence is that:

$$c(D_{R_t}) \leq c(D)/2$$

Remember that G is connected and simple.

Definition 3 (Hyperedge): *Let* $S \subseteq V$ *be a subset of nodes in G.* e_S *is a hyperedge defined on S if it covers the node set S.*

Sub-graphs of G can be associated with a hyperedge. Trivially, only sub-graphs spanning the nodes in S are interesting. Among spanning sub-graphs at most one with minimal cost exists: it is the minimum Steiner tree of S.

Definition 4 (Cost of a hyperedge): *We propose the association of the minimum Steiner tree with the hyperedge (if there are several trees, one of them can be selected arbitrarily.) We consider the cost of the minimum Steiner tree of S as the cost* $c(e_S)$ *of* e_S.

Definition 5 (k-limited hyperedge): *It is a hyperedge of size (cardinality) at most k*.

Hyperedges can be used to cover node sets. They can be isolated or connected.

Connectivity. To ensure connectivity, the hyperedges must be connected; there must be a path connecting each node in S to the other nodes via the hyperedges.

Definition 6 (Coverage): *A node set S is covered by a set of hyperedges, iff each node in S belongs to at most one hyperedge and the connectivity is true*.

Definition 7 (Cost of a coverage): *It is the sum of the costs of the hyperedges that compose it*.

We are interested in a coverage using "good" sets of hyperedges: two hyperedges in the coverage share at most one leaf, and there is no redundancy (cycle) in the set of hyperedges. Consequently, the coverage is a chain of hyperedges (a path) or a hypertree.

An important parameter of the coverage is the size k of the used hyperedges. The cost of the coverage and the complexity of the construction depend on it.

On the one hand, the higher the value of k, the closer the cost of coverage gets to the optimal coverage cost. For instance, if $k = |S|$ only one hyperedge covers the set S, and the cost is the cost of the minimum Steiner tree of S. It is the optimum.

On the other hand, the higher the value of k, the greater the computation time and cost. Remember, that the corresponding Steiner tree computation is exponential.

For this reason, only limited size hyperedges with small k values can be used in fast computations.

Often, to solve problems of covering sets of nodes, a metric closure graph is used, which contains the distances between the nodes. The distance of two nodes is the "cost" of the shortest path between them. Trivially, a shortest path is a hyperedge of size 2. The metric closure can be generalized based on larger (but limited) size hyperedges.

Definition 8 (Generalized metric closure): *Let M be a set of nodes in G. Let* $k < m = |M|$. *The set of hyperedges of size limited by k forms a k-limited metric closure*.

The generalized metric closure hypergraph contains the usual metric closure (the shortest paths), but also all of the hyperedges based on triplets of nodes, etc. Fig. 2 illustrates some hyperedges in the generalized metric closure of a small graph.

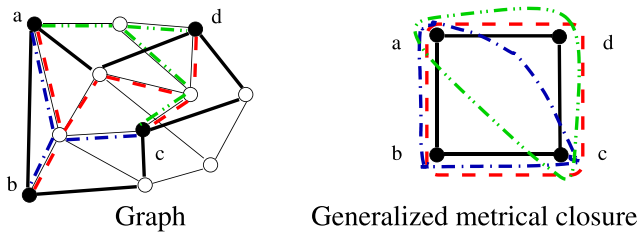

Figure 2: Some hyperedges in a generalized 3-limited metric closure.

To make the article easier to read, we summarize the main notations in Table 1.

Table 1: Notations.

$G = (V, E)$	Undirected Graph
V	Set of nodes in G
E	Set of edges in G
$L \subset V$	Set of leaves
L_i	Subset of leaves
m	Cardinality of L
$T = (W, F)$	Tree spanning L
T_i	Sub-tree
W	Set of nodes in T
F	Set of edges in T
$c(.)$	Cost function
P	Set of the paths in T
D	Diameter of T
R_i	i^{th} raceme in T
$S \subset L$	Subset of leaves
e_S	Hyperedge on S
k	Cardinality of S and e_S

4 Coverage of a Tree

In our particular first case, our objective is to cover a tree using k-limited hyperedges associated with leaves of the tree. Fig. 3 illustrates a hyperedge related to a set of leaves on a tree.

Let L be the set of leaves of a tree T.

Lemma 1: *Let $L1 \subseteq L$ be a subset of leaves and $S_k(L1)$ a set of hyperedges spanning L1. Let T1 be the sub-tree delimited by L1. The cost of $S_k(L1)$ is at most the length of the sub-tree T1.*

$$c(S_k(L1)) = \sum_{e_L \in S_k(L1)} c(e_L) \geq c(T1)$$

Proof: Each hyperedge corresponds to a sub-tree of $T1$. All of the leaves in $T1$ are covered. Suppose that the cost of $T1$ is greater than the sum of the costs of the hyperedges. In this case, there is at least an edge of $T1$ which is not included in any hyperedge. Since an edge is a separator for the set of leaves in a tree, the connection between the leaves of $T1$ by the hyperedges is not complete. The contradiction is trivial. □

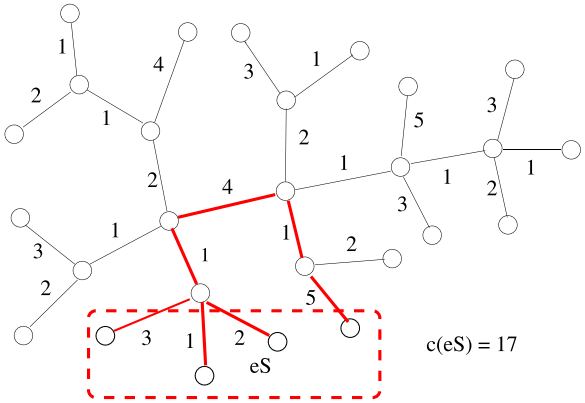

Figure 3: A hyperedge defined on a subset of leaves and its cost.

Consequently: if the set of hyperedges covers the leaves of T without exception, then the "sub-tree" delimited by them is T and the cost of the coverage is at least the cost of the tree T.

$$c(S_k(L)) = \sum_{e_L \in S_k(L)} c(e_L) \geq c(T)$$

Our analysis is limited to cases where the intersection of hyperedges belonging to a coverage of a tree T contains *only one node per peer*. The cost of the coverage can be computed as follows.

Some edges of the tree T are included in two hyperedges. Their cost is computed twice (once for each hyperedge) in the cost of the coverage. We refer to "duplicated" edges for cost calculation. Fig. 4 shows an example of the inclusion of some edges of T in the computation of the cost of two connected hyperedges. The duplicated edges form a path from the shared node in T.

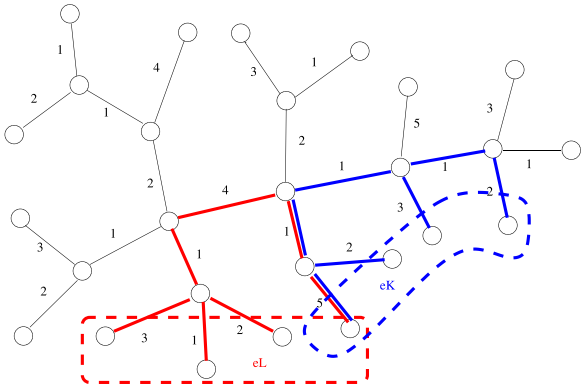

Figure 4: Two connected hyperedges of cardinality 4, the corresponding sub-trees and the duplicated edges of the tree.

Lemma 2: *Let* $L1$ *be a subset of leaves covered by a set of hyperedges. The cost of the coverage* $S_k(L1)$ *of* $L1$ *is the cost of the sub-tree delimited by* $L1$*, increased by the cost of the duplicated edges in the span. Let* D_{L_i,L_j} *be the set of duplicated edges for hyperedges* L_i *and* L_j*.*

$$c(S_k(L1)) = \sum_{e_L \in S_k(L1)} c(e_L) = c(T1) + \sum_{L_i \in S_k(L1), L_j \in S_k(L1), L_i \neq L_j} c(D_{L_i,L_j})$$

Proof: Trivial. □

Remember that we limit the coverage to cases where each intersection of hyperedges contains only one node. The examined hypergraphs are simple: the connected hyperedges can form a chain or a tree [21].

Lemma 3: *Let us suppose that* $m = |L|$. *The coverage needs* $n = \frac{m-1}{k-1}$ *hyperedges if only one node is shared between two connected hyperedges.*

Proof: n can be calculated recursively. If there is only one hyperedge, at most k leaves are covered:

$$n(1) = k$$

Adding a new connected hyperedge, the maximal number of newly covered leaves is $k-1$:

$$n(i) = n(i-1) + k - 1$$

To cover m leaves, $n = \dfrac{m-k}{k-1} + 1 = \dfrac{m-1}{k-1}$ hyperedges are needed. □

4.1 Construction of a Coverage

In this section, we first present a costly procedure for finding the best cost k-limited coverage. A second algorithm follows a greedy construction, without any guarantee regarding costs.

To find the coverage with minimal cost for an arbitrary value $2 < k < m$, different exact algorithms can be found. For instance, the following procedure can be used, which is based on the generalized metric closure and has two main steps.

1) For the set L of leaf nodes, construct the k-limited hyper metric closure $H(L)$ (cf. Algorithm 1).

For the integer values i from 2 to k, compute all i-tuples. Associate the cost of corresponding sub-trees with the hyperedges. Remember that there are $\binom{m}{i}$ tuples to create for a given i.

Algorithm 1: Hyper Metric Closure H(L)

Require: L, k
Ensure: $H(L) = (L, HE)$ ▷ The metric closure hypergraph
  $HE = \varnothing$
  for $i = 2$ to k **do**
    for each i-tuple $S_i \subset L$ **do**
      Compute the minimum Steiner tree ST of S_i
      $w = cost(ST)$
      Create a hyperedge he on S_i with cost w
      $HE \leftarrow HE \bigcup \{he\}$
    end for
  end for
    return $H(L)$

2) Find the set of connected hyperedges of minimal cost covering L. For this, the set of connected hyperedges covering L must be enumerated, and the best cost set must be selected.

This second problem (often called "FST concatenation phase" in the geoSteiner problem) is known [17]. Finding a subset of the hyperedges (corresponding to Steiner trees), whose concatenation produces the final Steiner tree, corresponds to a Minimum Spanning Tree problem in the hypergraph. This problem is NP-hard; for instance, a branch-and-cut approach or an exhaustive enumeration can be proposed to solve it (cf. Algorithm 2).

Algorithm 2: Optimal coverage with hyperedges

Require: $H(L) = (L, HE)$ ▷ The metrical closure
Ensure: H
  $W = \max_{H(L)} w$ ▷ The maximum of costs in *H(L)*
  $B = W * |L|$ ▷ A sufficiently big value
  for all $SH \subset HE$
    $wH = \sum_{SH} w$ ▷ The cost of the subset
    if (SH is connected) & (SH covers L) & ($wH < B$) **then**
      $B \leftarrow wH$
      $H \leftarrow SH$
    end if
  end for
    return H

The connected set of hyperedges can be a chain or a tree. It does not contain loops.

A fast solution can be based on a greedy strategy to construct a chain of hyperedges to cover the leaves of a tree T. Each hyperedge in the coverage contains (at most) k leaves (the last hyperedge can cover fewer leaves). Each intersection of hyperedges contains only one leaf node. Following Lemma 3 to cover the tree $\frac{m-1}{k-1}$ hyperedges are needed.

To create the chain, (for instance) the diameter D of the tree can be followed, and suitable hyperedges can be created successively. Fig. 5 illustrates a chain composed of hyperedges of cardinality 3. The algorithm is outlined as follows and detailed by Algorithm 3.

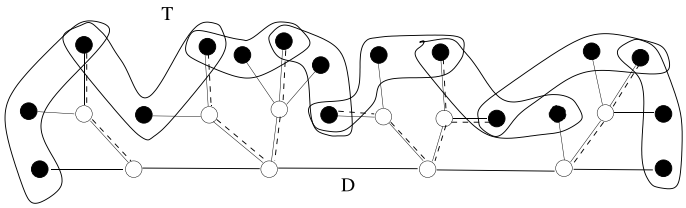

Figure 5: Illustration of a chain with $k = 3$. The pointed lines indicate the duplicated edges.

1) Starting from one of the endpoints of the diameter, a first leaf set is determined: the $k-1$ nearest leaves are added to the endpoint to form the set $S1$. A first hyperedge e_{S1} is created on the set $S1$ of leaves.

2) At each step of the suite, the sub-tree corresponding to the last hyperedge is examined. To create a new connected hyperedge, a leaf from this hyperedge is selected, and the $k-1$ nearest not yet covered leaves are added to this one to form the next set of leaves and the corresponding hyperedge. Since one leaf of the previous hyperedge belongs to the new hyperedge, a chain is created. The construction ends when all of the leaves of T are covered.

Algorithm 3: Greedy coverage with hyperedges

Require: $H(L) = (L, HE)$ ▷ The metrical closure
Ensure: HS ▷ A set of hyperedges
  $HS \leftarrow \varnothing$
  Compute D ▷ The diameter of T
  $p \leftarrow$ endpoint of D
  while L is not covered **do**
    $SN \leftarrow$ the $k-1$ nearest nodes to p
    Create a hyperedge H on $SN \bigcup \{p\}$
    $HS \leftarrow HS \bigcup \{H\}$
    $p \leftarrow$ the last node of SN
  end while
    return HS

Note that this greedy algorithm does not guarantee the best cost solution (which is not necessarily a chain, but can also be a tree). Furthermore, regarding chains, the choice of the starting point and the subsequent continuations influences the cost. Greedy decisions do not guarantee an optimal cost chain either. As indicated, the problem is NP-hard.

4.2 Performance of the Greedy Algorithm

To illustrate the performance of the greedy algorithm, we performed calculations on randomly generated trees. For each case, a tree with 70 nodes was created. Table 2 presents the results. Each line of the table corresponds to a hyperedge size and contains the average values of 10 computations (10 different trees were created for each size). The second column indicates the average number of leaves, and the third indicates the cost of the trees. The size of the covering hyperedges was varied, as it is indicated in the next column. For the different sizes, the average values of costs, numbers of nodes, and edges were registered. As expected, the coverage cost decreases when the size of the hyperedges increases. This coverage cost is always higher than the cost of the basic tree.

Table 2: Coverage of trees with hyper chains.

Nb of Nodes	Leaves	Tree Cost	Size k	Cost of the Hyperchain	Nodes in the Hyperedges	Number of Edges
70	35.9	104.3	4	200.7	142.4	133.8
70	35.1	104.6	5	195.3	139.2	132.3
70	35.3	103.6	6	170.5	120.4	114.4
70	35.9	103.8	7	170.6	118.8	113.5
70	35.8	104.1	8	162.9	113.6	108.9
70	33.4	103.4	9	158.7	106.6	104.6
70	35.9	104.2	10	157.6	108.7	105.7

Finding upper-bounds on costs is an important topic. The following lemma gives a light upper-bound related to costs.

Lemma 4: *The length of the duplicated part of T (the length of duplicated edges) is*

$$\sum_{L_i \in S_k(L1), L_j \in S_k(L1), L_i \neq L_j} c(D_{L_i,L_j})$$

s.t.

$$c(D_{L_i,L_j}) \leq D(R_t) \forall\, i,\ j = i+1$$

Proof: Trivial. There is no path from a leaf to the base node t longer than the diameter of the raceme (cf. the definition of the raceme diameter). □

For some particular trees, more interesting upper-bounds can be found. For instance, in spiders $\frac{k}{k-1}$ is an upper-bound. We leave the proof to the readers.

5 Approximation of Minimum Steiner Trees

As demonstrated in the literature, hypergraphs can be applied to model the Steiner problem. For instance, the optimal concatenation of full Steiner trees to obtain a minimum cost Steiner tree corresponds to finding a minimum spanning tree in a hypergraph (MSTH) [17,20]. The idea to cover/span the leaf nodes of a tree by k-limited hyperedges can be used to create approximations of minimum Steiner trees. In the Steiner problem, a set M of terminal nodes must be covered. That is, the set of "leaves" to be covered by the algorithm is the given terminal node set. Of course, the Steiner tree is not known, but the tree covering technique presented earlier can be applied. Subsets of M can be covered by hyperedges of limited size, and their concatenation leads to a tree spanning M.

Since the Steiner problem is NP-hard, finding the optimum for large instances is impossible within a reasonable timeframe. Feasible approximations can be obtained by constructing hypertrees or hyperchains on M.

For minimal cost Steiner trees, the following properties are true.

Property 2 (Sub-optimality): *Let ST be a sub-tree of a minimal cost Steiner tree T. Let stt and stl be the set of leaves and the Steiner terminal nodes in ST, respectively. ST is a minimal cost Steiner tree of $stt \bigcup stl$.*

Proof: Let us suppose that ST is not of minimal cost, and another minimal cost Steiner tree STO spanning $stt \bigcup stl$ exists. Next, by replacing ST with STO in T, we obtain a lower cost tree covering the terminal Steiner nodes. T is not with minimal cost; the contradiction is trivial. □

Inversely, a chain of PMSTs (minimum Steiner trees) is not necessarily a PMST.

Property 3 (Chain of PMSTs): *Let $ST1$ and $ST2$ be minimal cost Steiner trees (PMSTs) of node sets $st1$ and $st2$, respectively, such that they share a common terminal node: $|st1 \bigcap st2| = 1$. The union of $ST1$ and $ST2$ is not necessarily a minimal cost Steiner tree of $st1 \bigcup st2$.*

Proof: A simple negative example provides proof. Let a, b, and c be three Steiner terminal nodes. The shortest paths P_{ab} and P_{bc} are the PMSTs for $\{a, b\}$ and $\{b, c\}$, respectively. The concatenation of P_{ab} and P_{bc} is not obligatory the PMST of $\{a, b, c\}$. In some cases, this latter can contain a Steiner node with degree 3. □

The trees $ST1$ and $ST2$ can be associated with hyperedges defined on the node sets $st1$ and $st2$. The union of the trees corresponds to the chain of the hyperedges. As indicated in the previous sections, the chain (the union of the trees) can also contain duplicate edges, common nodes, and loops.

Even though a chain of PMSTs does not give a PMST, the technique can be used to obtain approximations.

5.1 An Approximate Steiner Tree Construction

Given a terminal node set M in a simple graph G, a partial spanning tree that covers M at a minimal/good cost is needed. Since the optimization is NP-hard, decomposing the set of nodes M into small subsets allows for the computation of smaller PMSTs using a reasonable computation time. In this computation, the size of the used hyperedges is limited by a value k, and the k-limited PMSTs in G are associated with them. The connected hyperedges span the $|M|$ terminal nodes.

An algorithm similar to the greedy algorithm proposed for the tree coverage can work as follows. Let k be the maximal size of subsets/hyperedges.

1) A first Steiner terminal set S is selected with k nodes. A first hyperedge e_{S1} is created on the set $S1$, and the k-limited PMST of $S1$ is computed.

2) At each step of the suite, the last hyperedge is examined. A leaf from this hyperedge is selected, and the next $k-1$ nodes from M are added to it to form the next hyperedge. Then the PMST of the separated node set is computed. Since one leaf of the previous hyperedge belongs to the new hyperedge, a chain is created. Construction ends when all nodes in M are covered (cf. Algorithm 4).

Algorithm 4: Approximate Steiner tree using hyperedges

Require: M in G, and k ▷ The node set
Ensure: HS ▷ A set of hyperedges
  $HS \leftarrow \varnothing$
  Select $p \in M$ ▷ A first node
  while M is not covered **do**
    $SN \leftarrow$ the $k-1$ nearest nodes to p
    Create a hyperedge H on $SN \bigcup \{p\}$
    $HS \leftarrow HS \bigcup \{H\}$
    $p \leftarrow$ the last node of SN
  end while
    return HS

3) The union of the resulting PMSTs is not necessarily a tree. It is a graph covering M, but it can contain duplicated edges and loops. The last step of the construction eliminates the useless edges, preserving the connectivity of the graph.

Trivially, the loop-free connected graph is a tree spanning/covering M.

The *complexity* of the computation is high.

The computation can be based on the metrical closure. For the Steiner set M of $m = |M|$ nodes, the k-limited hyper metric closure $H(M)$ can be computed in $O(\sum_{i=2,\ldots,k} \binom{m}{i} \cdot ST(i))$, where $ST(i)$ is the complexity of the computation of a Steiner tree of i nodes.

The greedy algorithm constructs a chain of hyperedges. To select the first node in the next hyperedge, $O(k)$ distances from the metric closure should be compared. Then $k-1$ other nodes can also be be selected using *HMC*. This selection can be made in $O(k \cdot m)$. To create the result $\frac{m-1}{k-1}$ hyperedges are needed. Finally, the complexity of this greedy algorithm is $O(\frac{m-1}{k-1} \cdot k \cdot m) \approx O(m^2)$.

In real-world cases, the construction of the metric closure is predominant.

Furthermore, memory usage depends primarily on two factors: size m and k. The metrical closure needs the storage of $\sum_{i=2,\ldots,k} \binom{m}{i}$ Steiner trees. Each Steiner tree can contain $O(|E|)$ edges. The result contains $\frac{m-1}{k-1}$ hyperedges. Their size is also bounded by $O(|E|)$.

5.2 Tests, Experimental Results

The performance of the proposed algorithm has been tested in random graphs and also in a known network topology for randomly generated terminal nodes. The PMSTs have been computed using the Gurobi optimizer.

In a first series, the graphs were generated by the Albert-Barabási algorithm [22]. (Often these graphs are called BA graphs.) As it is known, this algorithm produces random scale-free networks. Several natural and human-made systems, including the Internet, the World Wide Web, citation networks, and some social networks are thought to be approximately scale-free. In these typical "network" topology, certain nodes (called hubs) have high degree.

In BA-type random graphs of 80 nodes and 238 arcs, random terminal groups of 15 nodes have been generated. The terminal node sets have been covered by hyperchains. The size of the hyperedges has been varied between $2 \leq k \leq 8$, and the k-limited PMSTs have been computed. As references, the optimal PMSTs have also been computed. For each line (for each value of k), 10 random graphs with random groups have been created. The values in the table are the average values of 10 runs. Table 3 indicates the parameters of the exact PMSTs and those of the chains. The cost of the duplication and the number of duplicated graph edges are also indicated. The value of the reduced graph obtained after the elimination of loops follows this information. The last columns indicate the ratios between the graphs obtained (before and after reductions), and the basic PMSTs.

Table 3: Approximations of the PMST, first series.

	PMST			Chain			Dup Cost	Dup Edge nb	Red.Cost	Ratio	Red.Ratio
k	Cost	Nodes	Edges	Cost	Nodes	Edges					
2	42.7	25.6	24.6	76.10	50.9	43.90	20.9	12.00	55.2	1.79	1.30
3	41.2	25.4	24.4	68.10	45.3	40.63	15.7	10.27	52.4	1.65	1.27
4	44.8	26	25	70.20	42.8	38.97	15.8	9.84	54.4	1.57	1.22
5	40.7	24.8	23.8	57.70	37.1	34.45	9.6	6.71	48.1	1.42	1.18
6	41.6	24.5	23.5	58.43	35.8	32.95	8.7	5.25	49.8	1.41	1.20
7	40	24.7	23.7	46.40	30.8	28.80	4.6	3.57	41.8	1.17	1.06
8	40.9	24.5	23.5	50.30	31.2	29.20	3	2.14	47.3	1.23	1.15

In a second case, random BA graphs with 120 nodes and 328 arcs have been created, and groups of 35 terminals have been generated. Table 4 contains the results.

Table 4: Approximations of the PMST, second series.

	PMST			Chain			Dup Cost	Dup Edge nb	Red.Cost	Ratio	Red.Ratio
k	Cost	Nodes	Edges	Cost	Nodes	Edges					
2	83.9	52.9	51.9	196.4	138	121.06	69.8	48.05	126.6	2.34	1.51
3	82.3	50.8	49.8	175	118.1	107.46	50.5	36.00	124.5	2.13	1.51
4	82.7	52.2	51.2	157.7	106.7	98.59	40.7	30.36	117	1.91	1.41
5	82.6	50.5	49.5	148.7	95.7	89.60	34.3	22.76	114.4	1.80	1.38
6	85.6	54.9	53.9	144.4	96.2	90.15	31.7	22.43	112.7	1.69	1.32
7	81.4	52.1	51.1	130.3	85.4	81.29	24.4	17.86	105.9	1.60	1.30

(Continued)

Table 4 (continued)

k	PMST Cost	PMST Nodes	PMST Edges	Chain Cost	Chain Nodes	Chain Edges	Dup Cost	Dup Edge nb	Red.Cost	Ratio	Red.Ratio
8	84	53	52	135.8	87.1	82.03	24.9	17.81	110.9	1.62	1.32
9	83.6	53.3	52.3	125.6	82.3	78.64	19.8	14.89	105.8	1.50	1.27
10	84.8	51.8	50.8	126.3	77.7	74.15	22.1	14.44	104.2	1.49	1.23
11	84.4	53	52	122.2	77	73.10	21.1	13.40	101.1	1.45	1.20
12	80.7	51.5	50.2	114.6	73.1	70.37	15.8	11.31	98.8	1.42	1.22
13	88.8	54.6	53.6	120.5	76.9	73.58	15.2	11.03	105.3	1.36	1.19

A thirds series has been realized using a known, relatively small network topology: the GEANT network, illustrated in Fig. 6 [23]. In this connected graph of 40 nodes and 61 edges, random costs between 1 and 5 were assigned to the edges. For each value of k between 2 and 6, 10 random groups have been selected. (Since the group size was 8, larger hyperedges are not of interest.) The average values are shown in Table 5.

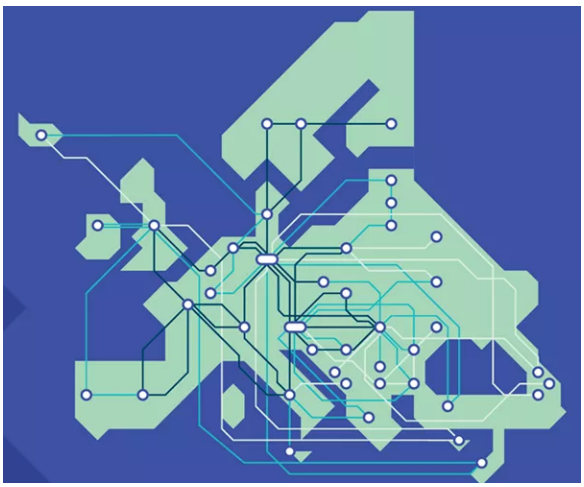

Figure 6: The GEANT network (source geant.org).

Table 5: Approximations of the PMST in the GEANT network.

k	PMST Cost	PMST Nodes	PMST Edges	Chain Cost	Chain Nodes	Chain Edges	Dup Cost	Dup Edge nb	Red.Cost	Ratio	Red.Ratio
2	31.5	15.3	14.3	55.8	27.9	23.9	16.1	7	39.7	1.77	1.25
3	33	16.6	15.6	49.8	26.5	23.5	10.3	5.4	40.3	1.51	1.23
4	29.9	15	14	37.7	18.9	16.9	4.8	2.3	32.9	1.27	1.11
5	33.6	15.8	14.8	41.9	20.6	18.6	4.1	2.3	37.8	1.25	1.12
6	31.9	15.7	14.7	41.9	20.8	18.8	5.3	2.7	36.6	1.32	1.15

To test more important trees in a bigger topology, we used a random graph with 1500 nodes and 5968 edges (Table 6). In this graph, PMSTs of groups with 300 nodes were computed. The coverage of the PMSTs was achieved using k-limited hyperedges with k = 10, 20, 30. This test clearly shows the effect of k.

Table 6: Approximations of the PMST in a random graph with 1500 nodes.

	PMST			Chain			Dup Cost	Dup Edge nb	Red.Cost	Ratio	Red.Ratio
k	Cost	Nodes	Edges	Cost	Nodes	Edges					
10	899.1	445.5	444.5	1694.1	930.6	900.5	502.6	347.1	1191.5	1.89	1.33
20	879.5	443.3	442.3	1137.25	788.9	773.9	344.5	236.8	1141.1	1.69	1.30
30	896.8	447.1	446.1	1403.1	740.6	730.6	296.7	208.7	1106.4	1.56	1.23

By increasing the size k of the hyperedges, the cost of the approximate solution decreases to a greater or lesser extent (the graphs are random, and each line represents the average values of 10 different graphs). In short, using larger hyperedges reduces the number of duplicated edges, which decreases the cost of the duplication.

An interesting result is that reducing the resulting graph (i.e., eliminating unnecessary duplication) greatly improves the algorithm's efficiency. The experimental ratio between the chain cost and the PMST is between 1.2 and 2.34, while the ratio between the reduced tree and the PMST is between 1.11 and 1.51. Duplicate edges can always be removed after calculating the small Steiner trees corresponding to the hyperedges.

6 Conclusions

In the first part of this work, we propose to cover a set of leaves of a tree with a set of connected hyperedges (with a hyperchain). Analysis of explicitly given trees shows that the cost of hyperchains covering the nodes is greater than the tree cost itself. This is a trivial result because some edges of the graph are used several times in the set of hyperedges. The idea of covering a tree with a set of hyperedges can be applied to approximately solve the Steiner problem, where only the terminal node set is known. A simple greedy algorithm is proposed to construct a hyperchain covering the Steiner terminal sets. The union of the small Steiner trees corresponding to the hyperedges of the covers gives a multigraph that can be reduced by eliminating loops corresponding to parallel edges. Experimental results show that the partial spanning tree obtained is a good approximation of the minimal Steiner tree. Unfortunately, at the moment, there is no theoretical result on the approximation ratio varying the hyperedge sizes.

In perspective, we suggest that the greedy algorithm can be improved. During the construction of chains, node groups could be selected more advantageously (for example, based on the distance between nodes). Furthermore, the construction of hypertrees covering the terminal nodes could also be explored. In-depth studies are needed to obtain eventual theoretical results.

Acknowledgement: Not applicable.

Funding Statement: The author received no specific funding for this study.

Availability of Data and Materials: Random graphs were generated using an internal code of graph generator. The data describing the parameters are available from the author upon reasonable request.

Ethics Approval: Not applicable.

Conflicts of Interest: The author declares no conflicts of interest.

References

1. Hwang F, Richards D, Winter P. The Steiner tree problem. In: Annals of discrete mathematics. Amsterdam, The Netherland: North-Holland; 1992.
2. Karp RM. Reducibility among combinatorial problems. In: 50 years of integer programming 1958-2008. Berlin/Heidelberg, Germany: Springer; 2010. p. 219–41.
3. Vazirani VV. Approximation algorithms. Berlin/Heidelberg, Germany: Springer Publishing Company, Incorporated; 2010.
4. Zelikovsky AZ. An 11/6-approximation algorithm for the steiner problem on graphs. In: Neŝetril J, Fiedler M, editors. Annals of discrete mathematics. Amsterdam, The Netherlands: Elsevier; 1992. p. 351–4.
5. Robins G, Zelikovsky A. Tighter bounds for graph steiner tree approximation. SIAM J Discret Math. 2005;19(1):122–34. doi:10.1137/S0895480101393155.
6. Byrka J, Grandoni F, Rothvoss T, Sanità L. Steiner tree approximation via iterative randomized rounding. J ACM. 2013;60(1):6. doi:10.1145/2432622.2432628.
7. Ljubic I. Solving Steiner trees: recent advances, challenges, and perspectives. Networks. 2021;77(2):177–204.
8. Su K, Lu B, Ngo H, Pardalos P, Du DZ. Steiner tree problems. In: Encyclopedia of optimization. Cham, Switzerland: Springer International Publishing; 2024. p. 1–15.
9. Rehfeldt D, Koch T. Implications, conflicts, and reductions for steiner trees. In: Proceedings of the 22nd International Conference on Integer Programming and Combinatorial Optimization; 2021 May 19–21; Atlanta, GA, USA. Cham, Switzerland: Springer; 2021. p. 473–87.
10. Rehfeldt D, Koch T. Implications, conflicts, and reductions for Steiner trees. Math Program. 2023;197(2):903–66. doi:10.1007/s10107-021-01757-5.
11. de Figueiredo CMH, Lopes R, de Melo AA, Silva A. Parameterized algorithms for Steiner tree and (connected) dominating set on path graphs. Networks. 2024;84(2):132–47. doi:10.1002/net.22220.
12. Hauptmann M, Karpinski M. A compendium on steiner tree problems. 2015 [cited 2026 Apr 16]. Available from: https://theory.cs.uni-bonn.de/info5/steinerkompendium/netcompendium.html.
13. Raikwar H, Karmakar S. Fully dynamic algorithm for the steiner tree problem in planar graphs. In: Proceedings of the 2022 Tenth International Symposium on Computing and Networking Workshops (CANDARW); 2022 Nov 21–24; Himeji, Japan. p. 416–20.
14. Raikwar H, Sadharakiya H, Karmakar S. Dynamic algorithms for approximate steiner trees. Concurr Comput Pract Exp. 2025;37(6–8):e70040. doi:10.1002/cpe.70040.
15. Balev S, Pigné Y, Sanlaville E, Vernet M. The dynamic steiner tree problem: definitions, complexity, algorithms. In: Proceedings of the 3rd Symposium on Algorithmic Foundations of Dynamic Networks (SAND 2024); 2024 Jun 5–7; Patras, Greece. doi:10.4230/LIPIcs.SAND.2024.24.
16. Balev S, Sanlaville E, Schoeters J. Temporally connected components. Theor Comput Sci. 2024;1013(3):114757. doi:10.1016/j.tcs.2024.114757.
17. Warme DM. Spanning trees in hypergraphs with applications to Steiner trees [dissertation]. Charlottesville, VA, USA: University of Virginia; 1998. doi:10.5555/927353.
18. Kaufmann M, van Kreveld M, Speckmann B. Subdivision drawings of hypergraphs. In: Tollis IG, Patrignani M, editors. Graph drawing. Berlin/Heidelberg, Germany: Springer; 2009. p. 396–407.
19. Brazil M, Zachariasen M. Steiner trees in graphs and hypergraphs. In: Optimal interconnection trees in the plane: theory, algorithms and applications. Cham, Switzerland: Springer International Publishing; 2015. p. 301–17.
20. Polzin T, Daneshmand SV. On Steiner trees and minimum spanning trees in hypergraphs. Oper Res Lett. 2003;31(1):12–20. doi:10.1016/s0167-6377(02)00185-2.
21. Ouvrard X. Hypergraphs: an introduction and review. arXiv:2002.05014. 2020.
22. Albert R, Barabási AL. Statistical mechanics of complex networks. Rev Mod Phys. 2002;74(1):47–97. doi:10.1103/revmodphys.74.47.
23. GEANT network. 2012 [cited 2026 Apr 16]. Available from: https://topology-zoo.org/files/Geant2012.gml.