On the Approximation of Degree Constrained Spanning Problems in Graphs under Non Uniform Capacity Constraints

Miklós Molnár

On the Approximation of Degree Constrained Spanning Problems in Graphs under Non-Uniform Capacity Constraints

Miklós Molnár

LIRMM, University of Montpellier, CNRS, Montpellier, France
molnar@lirmm.fr

Abstract. Degree constrained spanning tree problems with minimum cost in graphs are known and have been analyzed in depth. Usually the degree constraints are due to limited definitive *budget* of the graph nodes. For these NP-hard problems, spanning trees are the solutions if they exist and the optimum can not be approximated within a constant factor. Recently, analyses have been published to solve the degree constrained spanning problem when the constraints represent a limited momentary *capacity* of the nodes. Different positive integer upper bounds are associated with nodes to limit their degree for *each visit*. Since finding the minimum cost degree constrained spanning structure is NP-hard, heuristics are needed. Previously the case of homogeneous degree constraint has been analyzed. This paper focus on the spanning problem with heterogeneous capacity-like degree bounds. Any solution of minimum cost corresponds to an earlier proposed generalization of the tree concept, *i.e.* to a hierarchy. We investigate on the conditions of its existence, on the computation by heuristics and on the possibility of its approximation.
Keywords: Graph theory, spanning problems, degree constrained minimum spanning tree, hierarchy, degree constrained minimum spanning hierarchy, inhomogeneous constraints, conditions for existence, heuristic, approximation

1 Introduction

Connected structures spanning the nodes in graphs with minimum cost are important in several domains, for instance for efficient broadcast / multicast communications in networks and for routing in micro-circuits. In our analysis we suppose undirected graphs. In the simplest cases, the set or a given sub-set of nodes should be spanned with minimum cost. It is well-known that the sub-graph which spans the set of nodes with minimum cost is a minimum spanning tree (MST), and several polynomial-time algorithms to find such a tree are known. The partial spanning with minimum cost (the Steiner problem in graphs) is NP-hard.

Some applications must meet additional constraints. Various constrained spanning problems have been analyzed in graphs (*cf.* examples in [16,5,18]). Here we are interested in the degree constrained spanning problem. Each node

$v \in V$ of the graph $G = (V, E)$ is assigned a positive integer value $D(v)$ which represents the maximum degree of the node in any spanning structure. This degree is potentially different from the degree of v in G indicated by $d(v)$. In the literature we can find several propositions to span the node set of a graph respecting *budget* type degree bounds [6] [4]. In these cases, nodes have limited budgets and can not exceed the given limits in the spanning structures (which are trees). For instance, when the bounds are uniform and equal to 2, the tree is a path, and the problem corresponds to the Hamiltonian path problem. Degree constrained spanning tree problems are hard to solve and unfortunately constant factor approximations do not exist for them [17].

It is not always possible to span the nodes using trees with respect to the degree constraints [3]. If the degree bound does not correspond to a definitive budget but to a momentary limited *capacity* of the node, more advantageous solutions can be found. For example, in optical networks, the capacity of duplication of the switches may be limited for each incoming light, but a wavelength can be reused several times in a switch when it returns to that one (cf. [15]). As a result, the optical broadcast/multicast route may be different from a spanning tree. In [2], the authors propose a special walk returning to some nodes when the use of spanning trees is not possible due to the absence of nodes which may have a degree greater than two in the spanning structure. The proposed walk visits the nodes by using a Depth First Search algorithm on a spanning tree. This walk visits nodes at most twice. If the constraints could be greater than 2, branching nodes are possible, and the spanning structure could be a hierarchy crossing certain nodes several times [13]. In the case where the capacity constraints are uniform in the node set, a hierarchy based solution and its approximation have been proposed [14]. The analysis of this NP-hard minimum spanning problem shows that a hierarchy based solution always exists and the minimum spanning hierarchy can be approximated. We propose here the analysis of cases where momentary capacity constraints are not uniform but heterogeneous. We are investigating the following questions:

- In which cases is it possible to span the set of nodes while respecting heterogeneous constraints?
- How to formulate and compute the optimum (with minimum cost)?
- Can the eventual solution be approximated?

To the best of our knowledge, our analysis is the first for non-uniform degree constrained spanning problems supposing capacity (and not budget-like) limitations. We demonstrate that, similarly to the case with uniform capacity bounds, the optimal solution of the problem is a *hierarchy* and we formulate necessary and sufficient conditions to find it. The minimum cost spanning hierarchy problem, similarly to the minimum constrained spanning tree problem, is NP-hard. In some cases the optimal solution can be approximated. Finding a good approximated solution is very profitable to several applications (*e.g.,* in networks).

In the following sections, we propose a quick presentation of the well-known and discussed degree constrained spanning tree problems, followed by the defi-

nition of hierarchies and the analyzed capacity constrained spanning problems (cf. Section 2). Necessary and sufficient conditions for a solution to exist are formulated in Section 3. The hardness of the problem is also presented and a heuristic algorithm to compute a good solution is proposed in Section 4 and Section 5 analyzes approximations. The performance of the proposed heuristic is illustrated in Section 6. The last section gives some perspectives.

2 Degree Constrained Spanning Problems

We target graph spanning problems, where a single connected structure should span the entire set of nodes while respecting constraints on node degrees. We have no further restriction on the spanning structure, which can be a connected sub-graph or another connected structure (*e.g.,* a trail, etc).[1]

2.1 Notations

The most important notations used in the paper are as follows.

$G = (V, E)$	: the undirected graph to span
V	: the set of nodes in G
E	: the set of edges in G
$\{n, m\} \in E$	: an edge between nodes n and m
$c(n, m)$	: cost associated with the edge $\{n, m\}$
$d(v)$	: the degree of the node $v \in V$ in G
$D(v)$	: a positive integer associated with the node $v \in V$; upper bound for the degree of the node occurrences
$V_m \subseteq V$	: the sub-set of nodes with degree bound m
v^i	: i-th occurrence of $v \in V$ in a hierarchy
$T = (P, F)$	: a tree
P	: the set of nodes in T
F	: the set of edges in T
M	: the highest degree bound
$\Theta(v)$	: the set of nodes in P corresponding to the node $v \in V$

We propose a clear distinction between two categories of degree constraints:

- budget-like constraints: in this case the total number of adjacent nodes is limited in the spanning structure and the node can not have more adjacent nodes than its limit,
- capacity-like constraints: the number of adjacent nodes is limited *for each visit* of the node in the spanning structure.

First, let us quickly analyze the problems, where the constraints on the degree of nodes are budget-like constraints and the nodes can participate to the covering until their budget is exhausted.

[1] In our approach, the spanning structure of a graph G is given by a homomorphic mapping which associates each node of G to at least one node of a connected graph H (*cf.* 2.3).

2.2 Minimum Spanning Trees under Budget-like Degree Constraints

If the objective is to cover a set of nodes with a connected structure while respecting budget-like degree constraints, any optimal solution, if it exists, is a tree. Since cycles are useless in the connected structure. Let us suppose that there are cycles in the solution. Deleting one edge from each cycle, the node set remains covered but the cost of the solution is smaller.

The basic problem was formulated as follows.

Definition 1 (Degree Constrained Minimum Spanning Tree - DCMST) *Given an undirected graph $G = (V, E)$ with nonnegative costs $c(e)$ on the edges $e \in E$ and with positive integer degree bounds $D(v)$ on the nodes $v \in V$. In the DCMST problem, the objective is to find a spanning tree in which the degree of any node v is at most $D(v)$ and the total cost is minimized.*

The degree bounds can be uniform or not in the node set. In homogeneous case of bounds, an integer B is given and the maximum degree of any node in the spanning tree is at most B. For instance, if $B = 2$, no node can have a degree more than 2, and any solution is a minimum Hamiltonian path. The degree constrained spanning tree problem is NP-hard, and a solution do not always exist.

In [17], the authors present the problem as a hard network design problem. Their analysis fits in the frame of a generic bi-criteria optimization where the first objective corresponds to the respect of the budget (degree) constraints, and the second to the minimization of the cost. The paper also specifies the sub-graphs to solve the problem. The investigated classes are spanning trees, Steiner trees and generalized Steiner trees. The authors prove the hardness of the proposed optimizations and give some negative results on the approximations. The DCMST problem is not in APX: there is no polynomial-time ρ-approximation algorithm minimizing the cost when respecting the degree constraint, even when the same upper bound is supposed for all of the nodes. The case with non-uniform degree constraints was also shown to be NP-hard using a reduction from the Traveling Salesman Problem (cf. [5]). It is known that it is not always possible to span the nodes using trees with respect of the degree constraints [3]. As observed in [5], a spanning tree solution of the degree bounded problem may exist iff $\sum_{v \in V} D(v) \geq 2 \cdot |V| - 2$. The partial spanning problem with non-uniform upper bounds cannot be approximated within $(2 - \epsilon, \rho)$ for any $\epsilon > 0$ and $\rho > 1$ [17]. For any $\epsilon > 0$ and $\rho > 1$, there is no polynomial-time $(\tau - \epsilon, \rho)$-approximation algorithm for this problem, where τ is the lower bound on the performance ratio of any algorithm for finding minimum Steiner trees. In unweighted graphs constrained by a uniform degree bound MB, Fürer and Raghavachari proposed an $(1, MB + 1)$-approximation algorithm for the degree constrained MST [9]. The paper [20] generalizes the result to weighted graphs.

When some nodes have a degree bound equal to 1, these nodes should be considered as leaves in the spanning structure. A special spanning problem with given leaf nodes can be formulated. The description of the terminal Steiner

problem is in [10] and approximation algorithms for the problem can be found in [8] [22]

Remember that the objective of the mentioned spanning problems is to cover the concerned nodes using a connected structure minimizing the cost while satisfying the degree constraints. *The constraints are budget constraints, and spanning trees are the cost optimal structures.* It means that a node belongs only once to the optimal solution. In the case of momentary capacity limitation of the nodes, and supposing that this capacity can be renewed, the optimal solution can return several times to the nodes. The model has been presented in [14] for uniform degree bounds. The relaxation of the supposition that the solution corresponds to a tree is beneficial, since the problem can be solved even if spanning trees satisfying the constraints do not exist.

An optimal solution always corresponds to a spanning hierarchy. Hereafter, the reader can find a brief review of the *hierarchies* corresponding to the minimum cost solutions.

2.3 Hierarchies

The generalization of the tree concept based on graph homomorphism permits an accurate definition [13]. To simplify, let us suppose undirected graphs but the definition for digraphs is trivial.

Definition 2 (Hierarchy) *Let $G = (V, E)$ be an arbitrary graph and let $T = (P, F)$ be a tree. Let $h : P \to V$ be a homomorphic function which associates a node $v \in V$ to each node $p \in P$. The application (T, h, G) defines a hierarchy in G.*

Since a node $v \in V$ can be associated with several nodes in P, a hierarchy can return several times to nodes and pass several times through edges: it corresponds to a "non-elementary tree" in the graph G. In hierarchies certain nodes are eventually branching nodes; they are the node occurrences corresponding to the branching nodes of the tree T. Figure 1 illustrates a hierarchy in an undirected graph. Some nodes of the graph (namely the nodes c and d) participate on different levels of the hierarchy shown in the labeled tree T. In the figure, the labels are identifiers of nodes in G. More details can be found in [13].

Hierarchies generalize trees. Trees are special hierarchies without repeating nodes (applying an injective mapping h), and therefore inherit the properties of hierarchies. Notice that the sub-graph of G corresponding to a hierarchy (the projection) can contain cycles in G but the hierarchy itself preserves the structure of a tree. In the following, we use the term *hierarchy* to reference the defined tree-like structure and we use the term *image of the hierarchy* for the sub-graph implicated in G. Hierarchies allow the exact definition of the degree constrained spanning problem with capacity-like constraints.

2.4 Node Capacity Constrained Spanning Problems

Definition 3 (Node Capacity Constrained Minimum Spanning Problems) *In a graph G positive costs $c(e)$ are associated with edges $e \in E$, and positive*

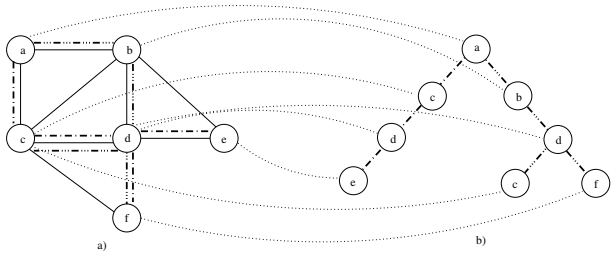

Fig. 1. Example of a hierarchy in a undirected graph

degree bounds $D(v)$ are associated with nodes $v \in V$. The bounds represent the instantaneous maximum capacity of the nodes. The problem consists in finding the minimum cost structure spanning the node set V, s.t. the degree constraints are respected for each visit of the nodes.

The solution of the problem has been identified in [13].

Proposition 1. *Any connected structure of minimal cost spanning a set of nodes in a connected graph under limited instantaneous capacities of the nodes is a hierarchy.*

Proof. To simplify, let us suppose that the solution exists (the reader can find the conditions in Section 3). Let the spanning structure be given by the homomorphism based triplet (H, h, G). To obtain a connected spanning structure, H should be a connected graph. Let us suppose that there exists an optimal solution that is not a hierarchy, but another connected structure covering the nodes and respecting the degree constraints for each visit of the nodes in G. In this case, H is not a tree but another connected graph. Therefor, it contains loops. Trivially, edges can be removed from loops without removing nodes and thus preserving the node coverage of G. The structure cannot be optimal in terms of cost. The connected graph H of minimum cost is a tree, so the triple corresponds to a hierarchy. ■

Solutions are Degree Constrained Minimum Spanning Hierarchies (DCMSHs), and the computation of this kind of solutions is the DCMSH problem. Figure 2 illustrates the interest of the hierarchies in the degree capacity constrained spanning problems. Let us suppose that the cost of any edge in the graph G is one. The different constraints on the node degrees are indicated in the figure. The minimum cost connected spanning of this graph is required in respect of the constraints. Trivially, such a spanning tree does not exist but there is a spanning hierarchy satisfying the constraints. It visits the node b twice, and each occurrence of this node respects the degree constraint as it is shown by the labeled tree T of the solution.

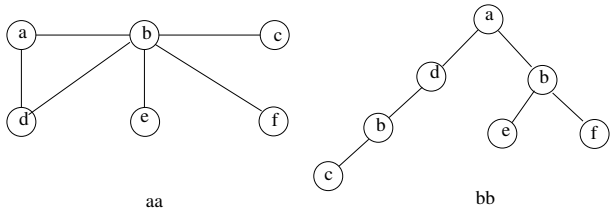

Fig. 2. A degree constrained minimum cost spanning hierarchy

The solution with uniform capacity bounds and its approximation have been presented in [14]. The approximation has been improved in [21]. An ILP based exact formulation of this problem can be found in [12]. The image of the solution in the case of homogeneous bounds was presented as a special trail: a k-trail for a degree bound equal to k [19].

The DCMSH problem is NP-hard.

As it was demonstrated in [14] spanning hierarchies satisfying a homogeneous degree constraint always exist. The following section discusses the conditions for the existence of a feasible solution in the case of non-uniform bounds.

3 Necessary and Sufficient Conditions for the Existence of a Solution

In this section, we show that a degree constrained spanning hierarchy does not always exist in the cases where degree bounds are heterogeneous.

Remember, a hierarchy in G is given by a triplet (T, h, G), where T is a tree. We use the notation $V_d \subseteq V$ to indicate the sub-set of nodes with degree bound d. For instance, in V_1 the degree bound is one, consequently the nodes in this set must be leaves in any spanning hierarchy.

Lemma 1. *If the specified leaf node set V_1 is empty, hierarchies spanning V and satisfying the degree constraints exist.*

Proof. A simple solution can be constructed as follows. Compute an MST in G and perform a walk in a deep first search manner. In this walk, each node occurrence has a degree at most two. The degree constraints are satisfied and the node set is covered. ■

Trivially, Lemma 1 gives a sufficient condition. The complementary case is when V_1 is not empty. Necessary and sufficient conditions for this case are discussed in the following.

Lemma 2. *If V_1 contains a separator[2], there is no hierarchy spanning V and satisfying the degree constraints.*

Proof. Let A and B be two non empty sub-sets of nodes separated by a separator $S \subseteq V_1$. Any path from a node in A to a node in B has to go through S but the nodes in S can not be internal nodes in any path. Trivially, there is no possible connection between nodes in A and nodes in B satisfying the degree constraints imposed by V_1. ■

Lemma permits to formulate a necessary condition: V_1 must be free of any separator. Note if V_1 is not a separator but a separator sub-set of V_1 exists, this later isolates another sub-set of nodes in V_1 which are unreachable from the remaining graph. Consequently, a spanning hierarchy satisfying the degree constraints does not exist. Figure 3 illustrates this case. Here, $V_1 = \{b, c, d\}$ is not a separator but the set $\{b, d\}$ is.

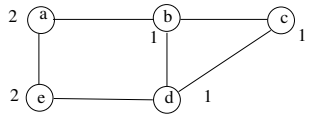

Fig. 3. V_1 is not a separator, but contains a separator

Lemma 3 and lemma 5 indicate that the set V_1 without separator is necessary but not sufficient for the existence of a spanning hierarchy satisfying the degree constraints. further sufficient conditions are given by Lemmas 4, 6, 7.

Lemma 3. *If $V_d = \emptyset, \forall d > 2$ (there is no node v with degree bound $D(v) > 2$) and $|V_1| > 2$ (there are more than two nodes with degree bound 1), there is no hierarchy spanning V and satisfying the constraints.*

Proof. A node $v \in V_1$ must be a leaf in the spanning hierarchy T, h, G (and trivially in the corresponding tree T). Since $|V_1| > 2$, the tree T and the hierarchy must have more than 2 leaves. Because $V_d = \emptyset, \forall d > 2$, the degree of internal nodes in T is equal to 2. T is a path and can not have more than two extremities. The contradiction is trivial. ■

Lemma 4. *If $|V_1| \leq 2$ and there is no separator in V_1, hierarchies spanning V and satisfying the constraints exist.*

[2] Remember that removing a separator from a connected graph, at least two connected components are obtained

Proof. Let us suppose that there are two nodes a and b with degree bound 1 and they do not compose a separator. A complete graph MC (*i.e.* the metrical closure) can be constructed in which the edges represent the shortest paths between the extremities. In MC, Hamiltonian paths exist whose endpoints coincide with a and b. Each Hamiltonian path corresponds to a Hamiltonian walk[3] in the original graph covering the node set and respecting the degree constraints. These walks are solutions. The same demonstration is trivially true with only one node with degree bound 1. In this case, in the metrical closure one end of the Hamiltonian paths can be freely chosen. ■

Lemma 5. *Let us suppose that* $|V \setminus V_1| = 1$ *in* G *and* $v_b \in V \setminus V_1$. *If the degree bound* $D(v_b) < |V_1|$, *there is no hierarchy spanning* V *and respecting the constraints.*

Proof. To avoid any separator in G, the nodes in V_1 should be connected to v_b. Suppose that a node $a \in V_1$ is not connected to v_b but to another node $b \in V_1$. b separates a from the other nodes. The nodes in V_1 must be leaves in any spanning hierarchy Moreover, there is no possible return from nodes in V_1 to v_b: there is only one occurrence of v_b in any hierarchy. This occurrence of v_b can have at most $D(v_b)$ neighbors. Since $|V_1| > D(v_b)$, all nodes in V_1 can not be connected to v_b in any spanning hierarchy respecting the constraints. ■

Lemma 6. *If* $|V \setminus V_1| = 1$, *the nodes in* V_1 *are neighbor nodes of the node* $v_b \in V \setminus V_1$, *and* $D(v_b) \geq |V_1|$, *a spanning hierarchy respecting the degree constrains exists.*

Proof. The graph G is a star. Since the degree bound of the central node $D(v_b) \geq |V_1|$, the star itself satisfies the degree constraints. ■

Lemma 3 and Lemma 5 indicate cases where there are several nodes in V_1 and there is no spanning hierarchy respecting the degree constraints. The following lemma formulates a condition to connect an arbitrary number of nodes in V_1 to a spanning hierarchy.

Lemma 7. *Suppose* G *has no separator in* V_1. *If* $|V \setminus V_1| \geq 2$ *(there are at least two nodes with degree bound greater than* 1*) and* $|V \setminus \{V_2 \cup V_1\}| \geq 1$ *(there is at least one node with degree bound greater than* 2*), a spanning hierarchy respecting the degree constrains exists.*

Proof. We prove that one node with degree bound two and another with degree bound greater than two are sufficient to construct spanning hierarchies for an arbitrary number of nodes in V. Let b be a node with $D(b) \geq 2$, and another

[3] A Hamiltonian walk is not obligatory an elementary Hamiltonian path and it can return several times to certain nodes

c with $D(c) > 2$. A degree constrained hierarchy spanning the node set can be constructed as follows. Because there is no separator in V_1, each node in $V \setminus (\{b\} \cup \{c\})$ can be connected to c with a path and each path respects the degree constraints. Then groups with these paths can be created s.t. there are $D(c) - 2$ paths in every group and in each group there is an occurrence of the node c. Each group corresponds to a spider respecting the degree constraints and having the occurrence of c as central node. The different occurrences of c can be chained by a walk visiting the node b (having a degree bound at least 2) s.t. the degree of each central nodes is at most $D(c)$. The obtained structure is a hierarchy spanning the node set and respecting the constraints. ■

Figure 4 illustrates a spanning hierarchy corresponding to Lemma 7. The occurrences of the node c (having a limit of degree equal to 3) are the central nodes of spiders which are connected by paths passing through the node b. Notice that it is not the "best" spanning hierarchy. The reader can easily find hierarchies containing less edges while respecting the constraints.

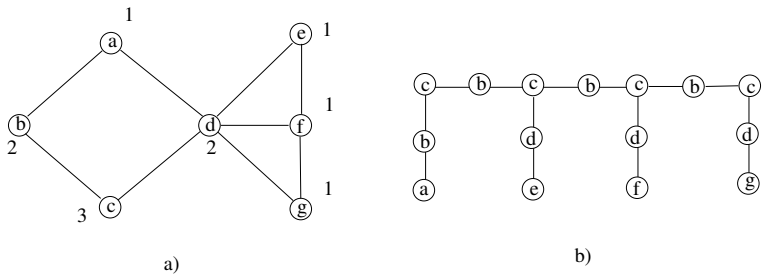

Fig. 4. Illustration of Lemma 7

The following theorem formulates necessary and sufficient conditions for the existence of degree constrained spanning hierarchies.

Theorem 1. *Let us name the conditions as follows:*
A: the desired leaf set V_1 *is empty*
B: there is no separator in V_1
C: $|V_1| \leq 2$ *(there are at most two nodes with degree bound* 1*)*
D: $|V_1| > 2$ *and the nodes in* V_1 *are neighbor nodes of a node* v *s.t.* $v \in V_m, m \geq |V_1|$
E: $|V \setminus V_1| \geq 2$ *(there are at least two nodes with degree bound greater than* 1*) and*
$|V \setminus (V_2 \cup V_1)| \geq 1$ *(there is at least one node with degree bound greater than* 2*).*

A hierarchy spanning all the nodes of a connected graph and respecting non-uniform capacity degree constraints can be found, iff

$$A \vee (B \wedge (C \vee D \vee E))$$

Proof. Following Lemma 1, condition A is sufficient.

If A is not true, the second part of the expression *i.e.* $B \wedge (C \vee D \vee E)$ gives a sufficient condition for the existence of a solution as it can be proved easily: in any case, V_1 can not contain a separator (condition B is necessary, as a consequence of Lemma 2) and one of the cases C,D or E must be true. Following Lemmas 4, 6 and 7 these conditions linked to B are sufficient conditions.

If V_1 is not empty and there is no separator in it, the expression $C \vee D \vee E$ also gives a necessary condition, that is one of them should be true for the existence. We prove that if none of them are true, there is no spanning hierarchy respecting the constraints. Let us suppose that none of them are true: there are more than two nodes in V_1 ($\neg C$), and they are not neighbors of a node with sufficiently large degree bound ($\neg D$), and there is no node with degree bound greater than 2 or another with degree bound 2 ($\neg E$). Start our prove by $\neg E$. If there is a single node with degree bound greater than 2, it must be the neighbor node of the nodes in V_1, but following $\neg D$, its bound is not sufficiently large. If there is no node with degree bound greater than 2, there are only nodes in V_1 and in V_2. Following $\neg C$, there are more than two nodes in V_1. Therefor the three conditions can not be false together (and another fourth condition can not guarantee the spanning hierarchy). ■

4 A Heuristic

Since the particular case of the problem with uniform degree constraint is NP-hard (*cf.* [12]), the more general problem with non-uniform constraints is NP-hard.

Several algorithms can be used to compute an optimal solution. With the modification of the ILP found in [12], an exact mathematical program of our problem can be obtained. For instance, branch and bound, branch and price like algorithms are candidates for the computation.

The approximation of the minimum spanning hierarchy under uniform capacity degree bound has been presented in [14] and improved in [21]. In the present paper, approximations under non-uniform degree bounds are analyzed, and a heuristic is proposed. The analysis of the conditions shows that the set of leaf nodes in V_1 strongly influences the solution. Sub-section 5.1 examines the approximation when the leaf node set is not fixed: $V_1 = \emptyset$. The problem is more complex in Sub-sections 5.2 and 5.3 when we suppose that the set V_1 is not empty. First of all, the computation of a lower bound on the cost of an optimal solution is presented and a heuristic algorithm is proposed (Sub-section 4.2).

4.1 A Lower Bound

The cost of a DCMST gives a trivial lower bound for the cost of any DCMSH: the DCMST is the degree constrained spanning structure with minimal cost.

Unfortunately this tree does not always exist and it can not be computed or approximated in polynomial time. Let us look at another spanning tree that can serve for a lower bound of the cost of DCMSHs. A particular minimum spanning tree can be built when a set V_1 of leaf nodes is given: these nodes must be leaves in the tree. It is the following.

Definition 4 (Minimum Spanning Tree with Specified Leaves - MSTSL) *Let $G = (V, E)$ be a connected graph, and let $V_1 \subset V$ be a non empty subset of desired leaves. Let us suppose that V_1 does not contain any separator. The degree of the other nodes is not limited in the spanning tree (for example $D(v) = d(v), \forall v \in V \setminus V_1$). The problem MSTSL consists in finding a minimum cost spanning tree covering the node set s.t. the nodes in V_1 are leaves in the tree.*

Notice that some other nodes in $V \setminus V_1$ can also be leaves in an MSTSL. A more complex study can be found in [7], which gives sufficient conditions for a graph to have a spanning tree with a specified set of leaves. It gives an Ore-type condition for a graph to be k-leaf-connected. [4] Algorithm 1 computes the MSTSL of a degree constrained graph.

Algorithm 1 COMPUTATION OF THE MSTSL

Require: A graph $G = (V, E)$, the set $V_1 \subset V$
Ensure: An MSTSL
  $V \leftarrow V \setminus V_1$ ▷ Delete V_1, the obtained graph is $G' = (V \setminus V_1, E')$
  $T' \leftarrow$ **MST** of G' ▷ *e.g.,* using Kruskal's algorithm
  for each $v \in V_1$ **do**
    $E_v \leftarrow$ set of adjacent edges of v with one extremity in V_1
    $e_v \leftarrow$ the edge with minimum cost in E_v
    $E' \leftarrow E' \cup \{e_v\}$ ▷ Connect v to T' by the edge e_v
  end for
  return T' ▷ The result is the MSTSL

Lemma 8. *Algorithm 1 computes an MSTSL in polynomial time.*

Proof. The resulted tree is an MSTSL. $G' = (V \setminus V_1, E')$ is connected, since V_1 does not contain any separator. T' is a tree which spans the whole node set. In T' the nodes in V_1 are leaves. Moreover T' is of minimum cost as it is proved in the following. Let us suppose that a tree T_m exists having a smaller cost and satisfies the constraints on the leaves. Let T'_m be the tree after the deletion of leaves in V_1 from T_m. In order to connect the leaves in V_1 to T'_m, trivially the edges having the smallest cost among the possibilities are used. Consequently,

[4] A graph $G = (V, E)$ is k-leaf-connected if $|V| > k$ and for each subset $S \in V$ with $|S| = k$, G has a spanning tree T with S as the set of leaves.

the cost of T'_m must be less than the initial cost of T' which contradicts the fact that the initial T' is an MST.

The result is computed in polynomial time. The initial MST T' is computed in polynomial time (*e.g.,* the complexity of Kruskal's algorithm is $\Theta(|E|\log|E|)$). The **for each** loop is executed at most $|V|-1$ times. Trivially, $|E_v| < |V|$. To find the minimum of edges $O(|E_v|)$ operations are needed. Finally the complexity of the second part (of the loop) is $O(|V|^2)$. ■

Lemma 9. *If the given node set V_1 contains a separator, the MSTSL does not exist. In the opposite case (there is no separator in V_1) the MSTSL exists.*

Proof. The first part is the direct consequence of Lemma 2. If V_1 contains a separator, there is no spanning hierarchy (and consequently spanning tree) satisfying the constrains. If there is no separator in V_1, by construction (*cf.* Algorithm 1), the MSTSL can be built. In the algorithm, after the deletion of V_1, in the remained connected graph the MST can be built and the reconnection of nodes to the MST is also possible. ■

Theorem 2. *Let us suppose that spanning hierarchies corresponding to the given degree constraints exist, satisfying the conditions in Theorem 1. Then an MSTSL with leaf nodes specified by V_1 exists and its cost is a lower bound for the cost of a DCMSH.*

Proof. Satisfying the conditions in Theorem 1 guarantees there is no separator in V_1 and the MSTSL exists (Lemma 9).

In a DCMSH H, each node in V_1 is a unique leaf occurrence (it is present only once). Let us suppose that a node in V_1 corresponds to two leaves in H. One of them can be deleted and the results spans the node set. Consequently, two occurrences of a node in V_1 can not be in a cost minimum solution.

By deleting the leaf nodes in V_1 from H, a hierarchy H' is obtained. The deletion involves the deletion of the set E_1 containing their adjacent edges (these edges are also present only once in the hierarchy). The image I' of H' in G may eventually contain cycles. Since a node of $v \in V \setminus V_1$ can be associated with several node occurrences in H and in H', it can have more than $D(v)$ neighbors in the image. I' does not necessarily respect the degree constraint. From each eventual cycle in I', an edge can be deleted and a tree T' in G covering the node set $V \setminus V_1$ is obtained (it is possible that T' does not respect the degree constraints either). By adding the deleted nodes in V_1 using the edges in E_1 to T' a spanning tree T_1 is created. It respects the constraints on the specified leaf nodes in V_1. Since T_1 contains the edges that are present in H only once and some edges are eventually deleted from the image I': $c(T_1) \leq c(H)$. Since an MSTLF T_L exists and T_1 is also a spanning tree respecting the constraints given by V_1, $c(T_L) \leq c(T_1)$.

Finally, $c(T_L) \leq c(T_1) \leq c(H)$. ■

4.2 The Proposed Heuristic

We propose a heuristic computing hierarchies providing an approximate solution to the DCMSH problem under non-uniform degree constraints. Since an MSTSL in G is a lower bound for a DCMSH and it can be computed in polynomial time, the heuristic computes an MSTSL. Then, this MSTSL is decomposed into a set of edge disjoint stars. The decomposition can be seen as a tree TS of stars in which each star (except the first) has a "parent". Each star is connected to its "parent" via its central node (which is a leaf in the "parent" star. An example of an MSTSL and its decomposition are shown in Figure 5. The degree bounds are indicated in the nodes. The corresponding global tree TS of the stars is also presented. A degree constrained spanning hierarchy with low cost is computed in each star of the decomposition. Finally, these spanning hierarchies are connected to form a unique hierarchy spanning the MSTSL (*cf.* Figure 6). Algorithm 2 shows the outline of the proposition.

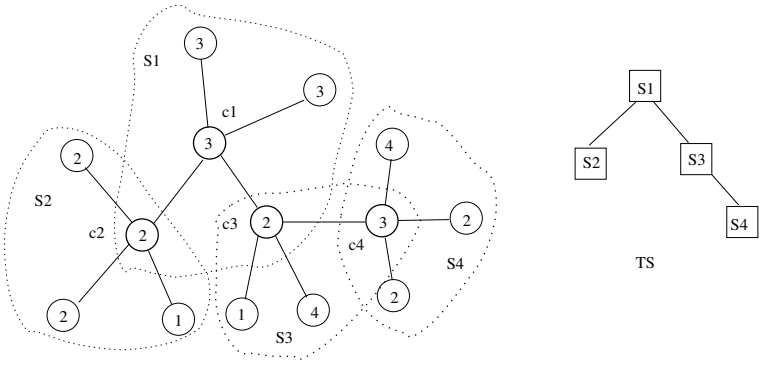

Fig. 5. The decomposition of an MSTSL

Decomposition of a tree into a set of edge disjoint stars Algorithm 3 presents a simple decomposition of a tree M into a set of edge disjoint stars. Let us suppose that there are more than three nodes in the tree (otherwise, the tree itself is a star and a decomposition is not needed). The decomposition can start with the unique neighbor node of an arbitrary leaf of M (*e.g.,* node c_1 in Figure 5). This node is considered as the central node of the first star and the nodes related to this central node are leaves in the star ($S1$ in the figure). If a leaf in the star is not a leaf node in M, it is considered as the central node of a following star. In our example, the nodes c_2 and c_3 are this kind of nodes in $S1$. The decomposition continues in the newly detected stars until there is no longer a leaf corresponding to a new center.

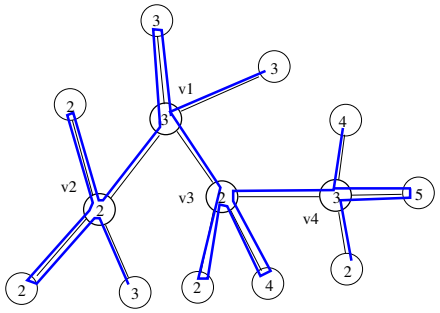

Fig. 6. The final hierarchy respecting the degree constraints

Algorithm 2 COMPUTATION OF AN APPROXIMATE DCMSH

Require: A graph $G = (V, E)$ with degree constraints
Ensure: A degree constrained spanning hierarchy
  $M \leftarrow$ **MSTSL** of G ▷ computation by Algorithm 1
  Decomposition of M into a set SS of disjoint stars
  ▷ computation by Algorithm 3
  for each star $S \in SS$ **do**
    $E_S \leftarrow$ **Adjacent edges** of node c in S ▷ c is the central node in S
    $E_P \leftarrow$ **Adjacent edges** of node c in P ▷ c is a leaf in the parent star P
    $d(c) \leftarrow |E_S|$ ▷ degree of the central node c in the star
    $dp(c) \leftarrow |E_P|$ ▷ degree of c in the parent star
    if $d(c) + dp(c) < D(c)$ **then**
      $S_c \leftarrow S$ ▷ the star corresponds to the constraints
    else
      $S_c \leftarrow$ **Degree constrained spanning hierarchy** for S
      ▷ computation by Algorithm 4
    end if
  end for
  Connect all hierarchies S_c to form a unique hierarchy H
  return H ▷ The result respects the degree constraints

Algorithm 3 DECOMPOSITION OF AN MSTSL

Require: An MSTSL $M = (V, E)$
Ensure: A set SS of edge disjoint stars
  $SS \leftarrow \emptyset$
  $L \leftarrow$ set of leaves in M
  $v \leftarrow$ a node from L ▷ an arbitrary leaf
  $w \leftarrow$ the neighbor node of v
  $C \leftarrow \{w\}$ ▷ set of possible center nodes of stars
  while C is not empty) **do**
    $c \leftarrow$ a node from C ▷ the next center node
    $N \leftarrow$ the neighbor node set of c
    $S \leftarrow$ a star with c, N ▷ the star centered by c
    $SS \leftarrow SS \cup S$
    $C \leftarrow C \cup (N \setminus L)$ ▷ eventual new center nodes
  end while
  return SS ▷ The set of edge disjoint stars

Computation of a degree constrained hierarchy spanning a star Remember, the decomposition of the MSTSL results in a set of edge disjoint stars. The crucial part of the solution is the computation of a degree constrained hierarchy in a star. In this section we propose a polynomial-time algorithm to span a star S with a hierarchy respecting the degree constraints. The outline of the computation is the following.

1) If $D(c) \geq d(c)$, the star itself satisfies the degree constraint; there is nothing to do.

2) If $D(c) < d(c)$ and there are more than two leaf nodes belonging to V_1 and there is no node with degree greater than two, there is no solution (*cf.* Theorem 1).

3) Otherwise, the star can be covered by a set $SC = \{Sc_i, i = 1 \dots n_S\}$ of chained small stars. These small stars are not edge disjoint but in the solution, the degree of each occurrence of c is limited by $D(c)$. An example of this chain of small stars in a star is illustrated in Figure 7. Notice that the dotted lines indicate the connections to the parent star (explained later).

All leaves in S are connected to c at least once and some are duplicated. Trivially, edges going to leaves in V_1 can not be duplicated. Our objective is to determine the set of duplicated edges to minimize the cost of the solution and to find a cost bound. Let E_{S1} the set of edges going to a node in V_1. To decide the "best" duplications, the edges not in E_{S1} are sorted in increasing order of costs in a list L. Let us indicate this first part of the list by L_F. Then the edges in E_{S1} are added at the end of the sorted list L. Let L_S be this second part of the list L. Based on this list, the chain of degree constrained stars can be organized as it is described in the following.

Starting at the first edge and going up in L_F, the best cost edges are selected for duplications. Each duplication of an edge $\{c, w\}$ adds a new occurrence of c to the hierarchy. Let $D(w)$ be the degree bound of w. $D(w) - 1$ returns from w

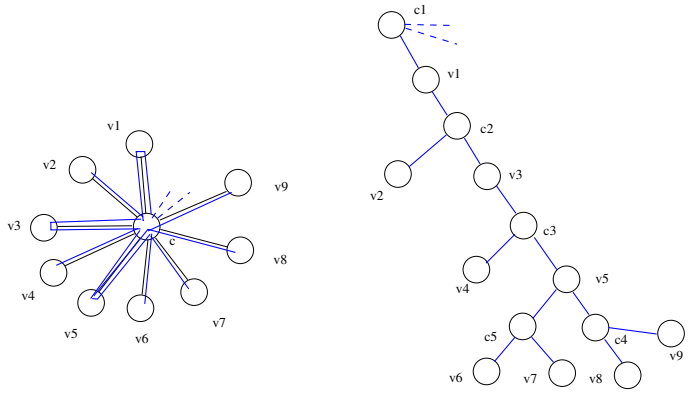

Fig. 7. Spanning hierarchy of a star computed by the proposed heuristic

are possible to small stars centered in new occurrences of c. Each small star can cover at most $D(c) - 1$ leaves different from w. In one of the new occurrences of c one adjacent edge should serve for the connection to the next neighbor star. In this way $(D(w) - 1) \cdot (D(c) - 1) - 1$ leaves different from w can be covered by the small stars centered on the new occurrences of c. Each newly covered leaf corresponds to an edge at the end of the list. This allows the coverage of the corresponding end nodes while respecting the degree constraints.

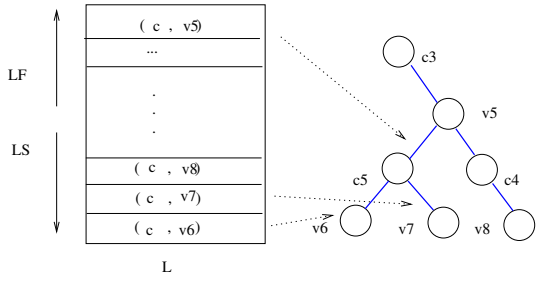

Fig. 8. One step of the algorithm

Figure 8 illustrates the selection of one edge for duplication from the list (and consequently the selection of edges found at the end of the list). In this example, $D(c) = 3$, v_5 corresponds to w, and $D(v_5) = 3$. This step results in the coverage of 4 leaves (including v_5).

In each step of the construction, the edges taken at the end of the list are deleted from the list. With moving forward in L_F, three cases are possible. If there are edges which are not yet selected from L_F, the construction continues as

described. If there is no more edges in L after the last selected one, the neighbor nodes of c are covered by the connected degree constrained stars, one can stop the construction. Otherwise (when L_S is not empty but the last edge selected for duplication is the last edge in L_F, the procedure continues by returning at the beginning of L_F.

The algorithm still needs a precision. The example in Figure 7 indicates the connections to the parent star. Except the first star ($S1$ in the decomposition in Figure 5) the hierarchy spanning a star must be connected via its central node to the hierarchy covering its parent star (the same node is a leaf in the parent star). The number of connections is decided by the construction of the parent hierarchy. For instance, the hierarchy covering $S2$ in Figure 5 must be connected via the node c_2 to the hierarchy spanning $S1$. In this example, the connection is a simple edge $\{c_1, c_2\}$.

The connections can be created as follows. The spanning hierarchy H_c of S_c with central node c must be connected to the hierarchy H_p spanning the parent star S_p (except the hierarchy in the first star). The connection is made using the node c which have a degree $dp(c) \leq D(c)$ in S_p. It is the number of adjacent edges of c in H_p. In the first star of the decomposition, $dp(c) = 0$. Suppose that c must have $d_c(c) > 0$ adjacent edges in H_c. To connect the $d_p(c) + d_c(c)$ neighbors to c, k chained occurrences of c are needed (the previous computation of k is not needed, the allocation of the nodes and edges is progressive with the presented algorithm). In the chain the first and the last occurrences of c can have $D(c) - 1$ adjacent edges, the middle occurrences can only have $D(c) - 2$ adjacent edges due to the edges used to chain the occurrences of centers. To connect all neighbor edges, $2 \cdot (D(c) - 1) + (k - 2) \cdot (D(c) - 2) \geq d_p(c) + d_c(c)$. Consequently, $k = \lceil \frac{d_p(c)+d_c(c)-2}{D(c)-2} \rceil$ occurrences are needed. The $d_p(c)$ edges in H_p can be connected arbitrarily to the occurrences of c (naturally the degree constraints must be respected). The $d_c(c)$ edges in H_c must be selected and connected to the occurrences of c from the previously described list. If $d_p(c) + d_c(c) \leq D(c)$, a unique occurrence of c can ensure the connections. If $d_p(c) < D(c)$, the first occurrence of c can serve to ensure the $dp(c)$ connections to the parent center node and the first $D(c) - dp(c) - 1$ edges in H_c. The remaining edges in H_c must be connected to the next occurrences of c in the defined order. If $d_p(c) = D(c)$, the first occurrence of c can have only $D(c) - 1$ adjacent edges in H_c and the last edge is connected to the second occurrence of c, etc.

In the example of Figure 7, $d_p(c) = 2$ and the first occurrence of c is connected twice to S_p and these connections are indicated with dashed line. Algorithm 4 gives the pseudo code of this construction.

Unfortunately the presented heuristic does not always work when a DCMSH exists. Theorem 1 summarizes conditions for the existence of a spanning hierarchy. To span a star with degree constrained hierarchies, these conditions must be met. If there is no node with degree bound greater than two in the star S and there are more than two leaves with degree bound one, S can not be spanned using a hierarchy respecting the degree constraints. That is, the proposed heuristic does not work in this kind of stars despite the fact that an eventual spanning

Algorithm 4 DEGREE CONSTRAINED SPANNING OF A STAR

Require: A star $S = (c \cup V_S, E_S)$, central node c, degree constraints, degree $dp(c)$ of c in the parent star

Ensure: A labeled tree $T = (P, F)$ corresponding to the solution (T, h, S)

$V_{S1} \leftarrow V_S \cap V_1$
$V_{S2} \leftarrow V_S \cap V_2$
if $(V_{S1} \neq \emptyset) \wedge (V_S \setminus (V_{S1} \cup V_{S2}) = \emptyset)$ **then**
  exit ▷ There is no solution
end if
$L_S \leftarrow \cup_{w \in V_{S1}}(c, w)$ ▷ List of edges going to nodes in V_{S1}
$L_P \leftarrow \textbf{sort}(E_S \setminus E_{S1})$ ▷ Sorted list of the other edges
$L \leftarrow L_P \cup L_S$ ▷ Concatenation of the two lists
$T \leftarrow c^1$ ▷ Initialize T with the first occurrence of c
if $D(c) = d_p(c)$ **then**
  $e \leftarrow \textbf{first}(L)$ ▷ Retrieve and delete the first edge $\{c, v\}$ of the list
  Connect v to c^1 in T using $\{c, v\}$ ▷ $D(c) - 1$ connections to parent node
  Connect c^2 to v in T using $\{c, v\}$ ▷ go and come
  $a \leftarrow D(c) - 1$ ▷ number of possible new neighbors of c^2
else ▷ c^1 can have the the adj. edges to the parent node
  $a \leftarrow D(c) - d_p(c)$ ▷ number of possible new neighbors of c^1
end if
$n \leftarrow 0$
while $(n < a) \wedge (L$ is not empty$)$ **do**
  $e \leftarrow \textbf{last}(L)$ ▷ Retrieve and delete the last edge $e = \{c, v\}$ of the list
  Connect v to c^1 in T using $\{c, v\}$
  $n \leftarrow n + 1$
end while
$n \leftarrow 0$
while $(n < D(c)) \wedge (L$ is not empty$)$ **do**
  if L_P in L is empty **then**
    $L \leftarrow L_P \cup L$ ▷ Add newly a copy of L_P to the remained list
  end if
  $e \leftarrow first(L)$ ▷ Retrieve and delete the first edge $e = \{c, v\}$ of the list
  Connect v to the last occurrence of c using $\{c, v\}$
  $m \leftarrow 0$
  while $[(n < D(c) - 1) \wedge (m < D(v))] \vee [(n = D(c) - 1) \wedge (m < D(v) - 1)] \wedge (L$ is not empty$)$ **do**
    $e \leftarrow last(L)$ ▷ Return and delete the last edge $\{c, v\}$ of the list
    Connect v to the last occurrence of c using $\{c, v\}$
    $m \leftarrow m + 1$
  end while
  $n \leftarrow n + 1$
end while
return T ▷ The labeled tree

hierarchy exists in the MSTSL. This latter hierarchy exists if there are nodes with degree greater than two in other stars in the MSTSL.

5 Approximations

Remember the DCMSH problem is NP-hard. In this section we give some ideas for its approximation. The approximation ratio found depends on the number and position of desired leaf nodes given in V_1. If there are a few numbers of nodes in V_1, guaranteed ratios can be found.

5.1 Case of $V_1 = \emptyset$

If $V_1 = \emptyset$, the cost of the DCMSH can always be approximated.

Theorem 3. *Let us suppose that there is no node with degree bound* 1 *($V_1 = \emptyset$). The cost of a DCMSH can be approximated within a ratio* $\frac{D}{D-1}$*, where* $D = \min_{v \in V} D(v)$.

Proof. An approximation scheme can be proposed starting from the MSTSL T of G. (In this case T is also the MST.) As it is demonstrated in the proposed heuristic, T can be decomposed into a set $S = \{S_i, i = 1, ...k\}$ of edge-disjoint stars. Let us recall that in each star S_i, a spanning hierarchy H_{Si}) respecting the degree constraint of the (eventually multiplied) central node c_i can be built. Since $V_1 = \emptyset$, the eventual duplications concern the best cost edges in the construction of H_{Si}). In the worst case the duplicated best cost edges are in V_2 and only one return is possible from the leaves. The cost of the hierarchy H_{Si} spanning the star S_i is bounded by $\frac{D(c_i)}{D(c_i)-1} \cdot c(S_i)$ where $c(S_i)$ is the cost of the star (*cf.* [14]). The hierarchies $H_{Si}, i = 1, \ldots, k$ spanning the k different stars can be connected, and a hierarchy H spanning the node set V and respecting the degree constraints is obtained. With $D = \min_{i=1,\ldots,k} D(c_i)$, since $\frac{D(c_i)}{D(c_i)-1} \leq \frac{D}{D-1}, i = 1, \ldots, k$, the cost of the spanning hierarchy is bounded:

$$c(H) = \sum_{i=1}^{k} c(H_{Si}) \leq \sum_{i=1}^{k} \frac{d(c_i)}{d(c_i) - 1} c(S_i) \leq \frac{D}{D-1} c(T)$$

■

In the worst case, the degree bound of all of the nodes is two, D is equal to two and the ratio of costs is also two. Trivially, the heuristic proposed in Section 4 provides this guarantee. If there are nodes with larger degree bounds, the heuristic computes hierarchies closer to the optimum.

5.2 Case of $0 < |V_1| \leq 2$

In this case, we suppose that V_1 is not empty but it contains at most two nodes which are not separators. We propose to analyze the case of $|V_1| = 2$, but the result is trivially true if there is only one node in V_1.

Theorem 4. *In the case of at most two beforehand specified leaves, the cost of a DCMSH can be approximated within a factor* 2.

Proof. Let a and b be the two specified nodes in V_1 and let T be an MSTSL having nodes a and b as leaves. Remember $c(T) \leq c(H)$, if H is a DCMSH. Following T, a walk[5] can be computed starting from a and ending at b. This walk W contains the edges of T at most twice and corresponds to a spanning hierarchy satisfying the degree constraints (the internal nodes in W have only degree 2).
$c(W) \leq 2 \cdot c(T) \leq 2 \cdot c(H)$. ■

Here too, sine the MSTSL is spanned by a walk, the ratio corresponds to the worst case. If there are nodes with degree bound greater than two, the heuristic proposed in Section 4.2 can compute better hierarchies with lower cost. Note that the heuristic does not always find a solution as it is indicated at the end of Section 4.2. It is the case when a and b are in a same star in the proposed decomposition and the star does not contain any node with degree bound greater then two.

5.3 Case of $|V_1| > 2$

Unfortunately, in arbitrary graphs with $|V_1| > 2$ an approximation with constant factor can not be guaranteed from an MSTSL.

Theorem 5. *Even if a DCMSH exists, it can not be approximated by a constant factor from the corresponding MSTSL when* $|V_1| > 2$.

Proof. Let us suppose a simple graph (corresponding to a star) as follows. Let a be a node with degree bound 3 and b the central node with degree bound 2. Let us suppose that k nodes with degree bound 1 are connected to the node b. Hence, a feasible solution exists. Consider the cost of the edges leading to leaves in v_1 being ϵ negligible to the cost of edge $\{a, b\}$. In this graph (cf. Figure 9), the MSTSL corresponds to the graph itself and its cost is equal to $1 + k \cdot \epsilon$. In the DCMSH (indicated in the figure), the possible branching nodes are occurrences of a. The cost of a path from a to an arbitrary leaf is $1 + \epsilon$. Since there are k leaves: $c(DCMSH) > k(1 + \epsilon)$ The ratio between the costs is bounded.

$$\frac{c(DCMSH)}{c(MSTSL)} > \frac{k(1+\epsilon)}{1+k\cdot\epsilon}$$

[5] A walk is a sequence of (eventually duplicated) edges which joins a sequence of nodes

When ϵ tends to zero, the lower bound tends to k. When k tends to infinity, the lower bound tends to infinity and can not be limited by a constant, valid for all stars. ■

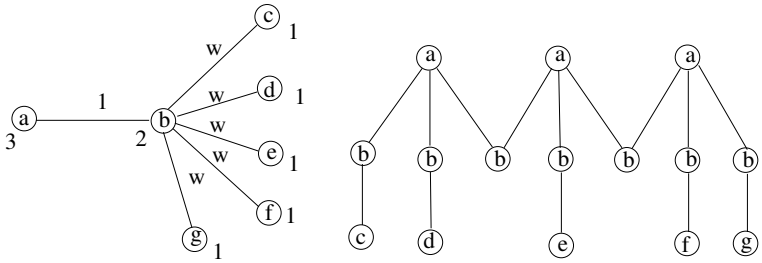

Fig. 9. A particular star and its DCMSH

In some particular graphs, approximations based on an MSTSL can be found. An example follows.

Lemma 10. *Let us suppose that* $|V_1| > 2$ *and a solution exists. Let* $V_c = \{v_c(i), i = 1, \ldots, m\}$ *be the set of nodes in the MSTSL which are neighbors of leaves in the MSTSL. Let* $V_c(i)$ *be the set of nodes in* V_1 *connected to* $v_c(i)$*. If* $D(v_c(i)) > |V_c(i)| + 1, \forall i$*, the DCMSH can be approximated from the MSTSL within a constant factor.*

Proof. Suppose the decomposition into a set of stars as in the proposed heuristic. The nodes in V_c are centers in stars in the decomposition. In each star, the nodes in $V_c(i)$ can be attached to one occurrence of $v_c(i)$ and the remaining star can be covered using further occurrence(s) of $v_c(i)$. More precisely, let us suppose that $L(i)$ is the set of leaves and $M(i) = L(i) \setminus V_c(i)$ is the set of leaves out of V_1 in the star. Let $C_s(i)$ and $C_c(i)$ be the cost of the star and the sum of the edge cost leaving to nodes in $V_c(i)$ respectively.

The cost of the sub star generated by $v_c(i) \cup M(i)$ is $C_m(i) = C_s(i) - C_c(i)$. This part can be covered respecting the degree constraint with an approximation ratio $\frac{d(v_c(i))}{d(v_c(i))-1}$ (*cf.* Theorem 3). The cost of the hierarchy spanning the star is:

$$C(H_i) \leq \frac{d(v_c(i))}{d(v_c(i)) - 1} C_c(i) + C_m(i) < \frac{d(v_c(i))}{d(v_c(i)) - 1} C_c(i) + \frac{d(v_c(i))}{d(v_c(i)) - 1} C_m(i)$$

$$C(H_i) < \frac{d(v_c(i))}{d(v_c(i)) - 1} C_s(i)$$

Similarly to the prove of Theorem 3, connecting the hierarchies spanning the different stars, a hierarchy H spanning the node set and respecting the degree

constraints is obtained. Let CS be the set of central nodes of stars ($V_c \subseteq CS$). With $D = \min_{c_i \in CS} D(c_i)$:

$$\frac{D(c_i)}{D(c_i)-1} \leq \frac{D}{D-1}, \forall c_i \in CS$$

If there are k stars in the decomposition:

$$c(H) = \sum_{i=1}^{k} c(H_i) \leq \sum_{i=1}^{k} \frac{d(c_i)}{d(c_i)-1} c(S_i) \leq \frac{D}{D-1} c(MSTSL)$$

■

Notice that the graph used in Theorem 5 shows an example where the MSTSL is the same as the MST and the attempt to find an approximation for the costs is based on the MSTSL. An approximation related to another subgraph (tree or spanning hierarchy computable in polynomial time) can be found or not is an open question.

6 Tests of the Heuristic

To illustrate the performance of the proposed heuristic, computations were performed in random graphs. The algorithms are implemented using the LEDA / C++ library [11]. A set of random graphs were generated using the Barabási-Albert model [1]. In this model, firstly a small connected graph (a kernel) is created and then new nodes are randomly connected to existing ones. In our tests, the kernel of the generated graph is a tree of 4 nodes and all graphs finally contain 100 nodes.

In each graph, the degree bounds of the nodes and the lengths of the edges have been randomly generated. The degree bounds have been randomly selected from the interval $[1, D_{max}]$ and the costs of edges from $[1, C_{max}]$. For the presented computations, the maximal cost has been set by $C_{max} = 10$, which allows sufficient variance of the costs. The value of D_{max} has been varied from 4 to 13. For each value of D_{max} 100 random graphs have been created. In the following table, each line corresponds to the average values of 100 runs (using 100 different graphs generated with the same parameters).

The first column shows the value of D_{max} followed by the average number $|E|$ of (randomly) generated edges and the average number $|V_1|$ of nodes with degree bound one. The next column indicates the number of graphs satisfying the conditions for the existence of the solution. The columns $C(MSTSL)$ and $C(Heur)$ contain the average cost values of the spanning trees with fixed leaves and the degree constrained spanning hierarchies respectively. The last column shows the ratio of these costs.

It is clear that as D_{max} increases, there are less nodes with degree bound one (the different degree bounds being equiprobable in the data generation).

	Average values					
Dmax	|E|	|V1|	cond %	c(MSTFL)	c(Heur)	c(MSTFL)/c(Heur)
4	373.46	25.96	85	178.588	228.765	1.28096
5	373.25	20.15	94	170.596	209.564	1.22842
6	374.02	16.13	97	166.701	192.340	1.15380
7	373.46	14.25	97	163.330	186.082	1.13930
8	373.74	12.82	98	165.082	187.786	1.13753
9	373.84	11.02	99	162.273	179.192	1.10426
10	374.03	10.00	100	161.180	178.700	1.10870
11	373.64	8.87	100	160.350	175.110	1.09205
12	373.30	8.36	100	159.320	168.860	1.05988
13	373.52	7.32	99	155.293	165.364	1.06485

Consequently the number of graphs in which the solution does not exist (because of the non satisfaction of the conditions for the existence of a solution) decreases. At the same time, the cost of the $MSTSL$ converges towards the cost of the MST and also the heuristic provides a solution with a cost value closer and closer to the cost of the $MSTSL$.

7 Conclusion and Perspectives

In our study, new results on degree constrained minimum spanning hierarchies in graphs with non-uniform capacity constraints are presented. As the solution does not always exist, necessary and sufficient conditions for the existence are formulated. Because the problem is NP-hard, a heuristic is proposed. In a set of random graphs, the heuristic provides results close of the optimum. The existence of approximation ratios is also raised. In some cases, the problem can be approximated. The set of specified leaves with degree bound 1 is crucial from this point of view. If this set is large, the solution may not exist and the approximation within a constant is an open question.

In future works, the approximation of spanning hierarchy problems with different degree constraints in various graphs should be analyzed largely and powerful algorithms should be investigated to compute good spanning hierarchies with guaranteed cost. We conjecture that in some cases, depending on positions of leaves in V_1, approximations may be found even if $|V_1| > 3$. Another important perspective is the analysis of partial spanning problems like degree constrained Steiner problems. We believe that a good part of the results presented in this paper can be applied in partial spanning cases.

Applications of the degree constrained spanning hierarchies can be found, for instance, in optical broadcast / multicast routing. In these applications, additional constraints (*e.g.,* the uniqueness of the use of a wavelength in a fiber, arc or edge in the graph the limited length of paths, etc;) exist, and the vari-

ous constraints should be fully satisfied. Analyzing these issues promises further interesting challenges.

References

1. Réka Albert and Albert-László Barabási. Statistical mechanics of complex networks. *Rev. Mod. Phys.*, 74:47–97, Jan 2002.
2. Maher Ali and Jitender Deogun. Cost-effective implementation of multicasting in wavelength-routed networks. *IEEE J. Lightwave Technol., Special Issue on Optical Networks*, 18(12):1628–1638, 2000.
3. F. Bauer and A. Varma. Degree-constrained multicasting in point-to-point networks. In *INFOCOM '95: Proceedings of the Fourteenth Annual Joint Conference of the IEEE Computer and Communication Societies (Vol. 1)*, page 369, Washington, DC, USA, 1995. IEEE Computer Society.
4. Bruce Boldon, Narsingh Deo, and Nishit Kumar. Minimum-Weight Degree-Constrained Spanning Tree Problem: Heuristics and Implementation on an SIMD Parallel Machine. *Parallel Computing*, 22(3):369–382, 1996.
5. Dietmar Cieslik. The vertex degrees of minimum spanning trees. *European Journal of Operational Research*, 125(2):278–282, September 2000.
6. N. Deo and S.L. Hakimi. The shortest generalized Hamiltonian tree. In *Sixth Annual Allerton Conference*, pages 879–888, 1968.
7. Yoshimi Egawa, Haruhide Matsuda, Tomoki Yamashita, and Kiyoshi Yoshimoto. On a spanning tree with specified leaves. *Graphs Comb.*, 24(1):13–18, 2008.
8. Bernhard Fuchs. A note on the terminal Steiner tree problem. *Inf. Process. Lett.*, 87(4):219–220, 2003.
9. Martin Fürer and Balaji Raghavachari. Approximating the minimum-degree Steiner tree to within one of optimal. *Journal of Algorithms*, 17:409–423, 1994.
10. Guohui Lin and Guoliang Xue. On the terminal Steiner tree problem. *Inf. Process. Lett.*, 84(2):103–107, 2002.
11. Kurt Mehlhorn and Stefan Näher. *LEDA: A Platform for Combinatorial and Geometric Computing*. Cambridge University Press, 1999.
12. Massinissa Merabet, Miklós Molnár, and Sylvain Durand. ILP formulation of the degree-constrained minimum spanning hierarchy problem. *Journal of Combinatorial Optimization*, 36(3):789–811, October 2018.
13. Miklós Molnár. Hierarchies to Solve Constrained Connected Spanning Problems. Technical Report 11029, LIRMM, July 2011.
14. Miklós Molnár, Sylvain Durand, and Massinissa Merabet. Approximation of the Degree-Constrained Minimum Spanning Hierarchies. In *SIROCCO*, pages 96–107, 2014.
15. Biswanath Mukherjee. *Optical WDM Networks (Optical Networks)*. Springer-Verlag, Berlin, Heidelberg, 2006.
16. Christos H. Papadimitriou and Mihalis Yannakakis. The complexity of restricted minimum spanning tree problems. In Hermann A. Maurer, editor, *Automata, Languages and Programming*, pages 460–470, Berlin, Heidelberg, 1979. Springer Berlin Heidelberg.
17. R. Ravi, Madhav V. Marathe, S. S. Ravi, Daniel J. Rosenkrantz, and Harry B. Hunt Iii. Approximation algorithms for degree-constrained minimum-cost network-design problems. *Algorithmica*, 31(1):58–78, 2001.

18. Stefan Ruzika and Horst W. Hamacher. *A Survey on Multiple Objective Minimum Spanning Tree Problems*, pages 104–116. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.
19. András Sebö, Alantha Newman, and Miklós Molnár. Travelling Salesmen on Bounded Degree Trails. In *Routing and Network Design Workshop*, Bonn, Germany, September 2015.
20. Mohit Singh and Lap Chi Lau. Approximating minimum bounded degree spanning trees to within one of optimal. In *STOC '07: Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 661–670, New York, NY, USA, 2007. ACM.
21. Mohit Singh and Rico Zenklusen. k-trails: Recognition, complexity, and approximations. In Quentin Louveaux and Martin Skutella, editors, *Integer Programming and Combinatorial Optimization*, pages 114–125, Cham, 2016. Springer International Publishing.
22. Fábio Viduani Martinez, José Coelho de Pina, and José Soares. Algorithms for Terminal Steiner Trees. In Lusheng Wang, editor, *Computing and Combinatorics*, pages 369–379, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.