# TWO DEPENDENCY CONSTRAINED SPANNING TREE PROBLEMS

**Luiz Alberto do Carmo Viana**
Universidade Federal do Ceará
www.ufc.br
`luizalberto@crateus.ufc.br`

**Manoel Campêlo**
Universidade Federal do Ceará
www.ufc.br
`mcampelo@lia.ufc.br`

## ABSTRACT

We introduce two spanning tree problems with dependency constraints, where an edge can be chosen only if *at least* one or *all* edges in its dependency set are also chosen, respectively. The dependencies on the input graph $G$ are described by a digraph $D$ whose vertices are the edges of $G$, and the in-neighbors of a vertex are its dependency set. We show that both problems are NP-hard even if $G$ is a chordal cactus with diameter 2 or maximum degree 3, and $D$ is a disjoint union of arborescences of height 2. We also prove that the problems are inapproximable to a $\ln n$ factor, unless $P = NP$, and that they are $W[2]$-hard. On the other hand, we present some polynomial cases. We test ILP formulations based on DCUT and MTZ constraints. Computational experiments are reported.

**KEYWORDS. Dependency constrained spanning tree. Computational complexity. Inapproximability.**
**Paper topics: OC. PM. TAG**

## RESUMO

Introduzimos dois problemas de árvore geradora com restrições de dependência, onde uma aresta pode ser escolhida apenas se *pelo menos uma* ou *todas* as arestas em seu conjunto de dependências também são escolhidas, respectivamente. As dependências no grafo de entrada $G$ são descritas por um digrafo $D$, cujos vértices são as arestas de $G$, e os vizinhos de entrada de um vértice são seu conjunto de dependências. Mostramos que ambos os problemas são NP-difíceis mesmo que $G$ seja um cacto cordal com diâmetro 2 ou grau máximo 3, e $D$ seja a união disjunta de arborescências de altura 2. Provamos também que os problemas são inaproximáveis por um fator $\ln n$, a menos que $P = NP$, e que eles são $W[2]$-difíceis. Por outro lado, apresentamos alguns casos polinomiais. Avaliamos formulações de programação inteira baseadas em restrições DCUT e MTZ. Experimentos computacionias são relatados.

## 1. Introduction

A spanning tree is a simple structure that recurrently appears in many applications. It describes, for example, a minimal subset of links of a network that keeps it connected with no redundancy. The basic problem in this context is the Minimum Spanning Tree Problem (abbreviated here as `MST`), which consists in finding a spanning tree of minimum cost. Although easily solvable in its basic version, this problem may became hard with the addition of extra requirements. This happens, for instance, if we require nonleaf vertices to have either a minimum or maximum degree.

Another NP-hard variation of `MST` consists in imposing conflict constraints over pairs of edges (Darmann et al. [2011]; Zhang et al. [2011]; Samer e Urrutia [2015]). A conflict between a pair of edges means that at most one of them may take part in the solution. These constraints are naturally described by an undirected simple graph, where the extremes of each edge represent a pair of conflicting edges of the input graph. Inspired by this problem, we introduce dependency constrained spanning tree problems, where dependency relations are represented by a directed graph. Basically, the occurence of an edge in the solution tree depends on the inclusion of other edges also in the tree.

Let $G = (V, E)$ be a connected graph and $D = (E, A)$ be a digraph whose vertices are the edges of $G$. We say $D$ is a dependency digraph for $E$, and $e_1 \in E$ is a dependency of $e_2 \in E$ if $(e_1, e_2) \in A$. The *Least-Dependency Constrained Spanning Tree* problem ($\texttt{L-DCST}(G, D)$) consists in deciding whether there is a spanning tree $T$ of $G$ such that each of its edges either has no dependency in $D$ or *at least* one of them is in $T$. Similarly, the *All-Dependency Constrained Spanning Tree* problem ($\texttt{A-DCST}(G, D)$) consists in deciding whether there is a spanning tree $T$ of $G$ such that each of its edges either has no dependency or *all* of them are in $T$. Note that the two problems coincide if $\Delta^-(D) = 1$, where $\Delta^-(D)$ is the maximum cardinality of an in-neighborhood of a node of $D$. In this case, we simplify the notation to $\texttt{DCST}(G, D)$.

The corresponding optimization versions, where a weighting function $w : E \to \mathbb{R}^+$ is considered and we want to minimize the weight of the spanning tree, are denoted respectively $\texttt{L-DCMST}(G, D, w)$, $\texttt{A-DCMST}(G, D, w)$, and $\texttt{DCMST}(G, D, w)$. Without loss of generality, we assume that the weights are non-negative because, if there were negative weights, we could choose a constant $s$ and shift $w'_e = w_e + s \geq 0$, for all $e \in E$, altering the solution cost by a constant factor $(n(G) - 1)s$, where $n(G)$ is the number of edges of $G$. Recall that the number of edges in any solution is $n(G) - 1$.

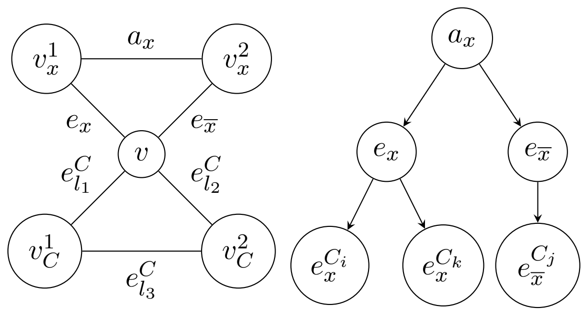
Applications for these problems appear, for instance, in communication systems when a link can only be used if the message arrives through certain other links, due to protocol conversion restrictions on the nodes of the network (Viana [2016]), or in trasportation netwotk projects where the construction of road depends on the construction of another one, and vice-versa.

## 2. NP-completeness

In this section we prove that problems $\texttt{L-DCST}(G, D)$ and $\texttt{A-DCST}(G, D)$ are NP-complete. In order to get strong hardness result, we restrict both $G$ and $D$ to have very simple structures. We use a reduction from `2 in 3 3-SAT`.

An NP-complete variation of `3-SAT` is `1 in 3 3-SAT` which consists in deciding whether a formula can be satisfied in such a way that every clause has exactly one true literal (Gary e Johnson [1979]). We can define `2 in 3 3-SAT` analogously. Note that `1 in 3 3-SAT` can be reduced to `2 in 3 3-SAT` by negating the literals of all clauses.

Given an instance of `2 in 3 3-SAT`, we build an instance of $\texttt{DCST}(G, D)$ as illustrated in Figure 1. We start $G$ with a universal vertex $v$; for each variable $x$, $v$ is connected to vertices $v_x^1$ and $v_x^2$ by edges $e_x$ and $e_{\overline{x}}$, respectively, and $v_x^1$ and $v_x^2$ are neighbors by edge $a_x$; for each clause $C = l_1 \vee l_2 \vee l_3$, $v$ is connected to vertices $v_C^1$ and $v_C^2$ by edges $e_{l_1}^C$ and $e_{l_2}^C$, respectively, while $v_C^1$ and $v_C^2$ are linked by edge $e_{l_3}^C$. We build $D$ as follows: there are arcs from $a_x$ to $e_x$ and $e_{\overline{x}}$, for each variable $x$; there is an arc from $e_l$ to $e_l^C$ if literal $l$ occurs in clause $C$. Note that $G$ is a chordal cactus (actually, a union of triangles whose pairwise intersection is $v$). $D$ is a union of arborescences and satisfies $\Delta^-(D) = 1$.



(a) Graph $G$.　　(b) Digraph $D$

Figure 1: Illustration of **SAT** reduction.

**Theorem 2.1.** *$\texttt{DCST}(G, D)$ is NP-complete, even if $G$ is a chordal cactus whose diameter is 2, and $D$ is a union of arborescences whose height is 2.*

*Proof.* Clearly, $\texttt{DCST}(G, D)$ is in NP. To prove that the reduction is valid, first take a valuation satisfying an instance of `2 in 3 3-SAT`. Consider the spanning subgraph $T$ of $G$ induced by the following edges: $a_x$, for every variable $x$; either $e_x$ or $e_{\overline{x}}$, depending on the valuation of $x$, for every variable $x$; for each clause $C = l_1 \vee l_2 \vee l_3$, the edges $e_{l_i}^C$ and $e_{l_j}^C$ such that $l_i$ and $l_j$ are true, $1 \leq i, j, \leq 3, i \neq j$. Observe that the edges of $T$ satisfy the dependencies imposed by $D$. We next show that $T$ is a tree.

For every variable $x$, $v$ is connected to either $v_x^1$ or $v_x^2$ by edges $e_x$ or $e_{\overline{x}}$, respectively, and the edge $a_x$ connects $v_x^1$ to $v_x^2$. Observe that there is no cycle among $v, v_x^1$ and $v_x^2$, and the two selected edges keep them connected. For every clause $C = l_1 \vee l_2 \vee l_3$, since two of the edges $e_{l_1}^C$, $e_{l_2}^C$ and $e_{l_3}^C$ are in $T$, we see that $v$ is connected to at least one of $v_C^1$ and $v_C^2$. Suppose that $v$ is connected to $v_C^1$ by $e_{l_1}^C$. If $e_{l_2}^C$ is in $T$, $v_C^2$ is also connected to $v$; otherwise, we have $v_C^2$ connected to $v_C^1$ by $e_{l_3}^C$. In both cases, there is no cycle among $v$, $v_C^1$ and $v_C^2$, and the two edges keep them connected. In this way, $v$ and the vertices associated to each variable induce a connected and acyclic subgraph of $T$. The same applies for the vertices related to each clause. We conclude that $T$ is a tree, and therefore a spanning tree of $G$.

Conversely, let $T = (V, E')$ be a feasible solution for $\texttt{DCST}(G, D)$. Note that, for every variable $x$, either edge $e_x$ or $e_{\overline{x}}$ is in $T$, since they are a cut and both depend on $a_x$. For every clause $C = l_1 \vee l_2 \vee l_3$, exactly two of the edges $e_{l_1}^C$, $e_{l_2}^C$ and $e_{l_3}^C$ are in $T$ connecting $v_C^1$, $v_C^2$ and $v$ without inducing a cycle. Since $e_l^C$ is in $T$ only if $e_l$ is in $T$, we valuate each variable $x$ as true ($e_x \in E'$) or false ($e_{\overline{x}} \in E'$), and this valuation satisfies every clause $C$ with exactly two true valued literals. In this way, we decide the corresponding instance of `2 in 3 3-SAT` is satisfatible. □

Notice that $G$ is planar and has arbitrary $\Delta(G)$, the maximum degree of $G$. We can rearrange its triangles to get $\Delta(G) = 3$. We make the triangles (related to clauses and variables) disjoint and link them as in Figure 2. In other words, we split vertex $v$ into a vertex $v_C$ for each clause $C$ and a vertex $v_x$ for each variable $x$. This modified reduction leads to:

**Theorem 2.2.** *$\texttt{DCST}(G, D)$ is NP-complete, even if $G$ is a chordal cactus with $\Delta(G) \leq 3$, and $D$ is a union of arborescences whose height is 2.*

## 3. Inapproximability

In this section, we show an inapproximability threshold for problems $\texttt{L-DCMST}(G, D, w)$ and $\texttt{A-DCMST}(G, D, w)$, when $w$ is a 0-1 function. We also show that they are $W[2]$-hard parame-

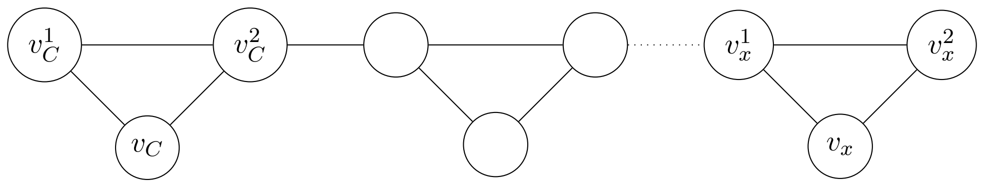
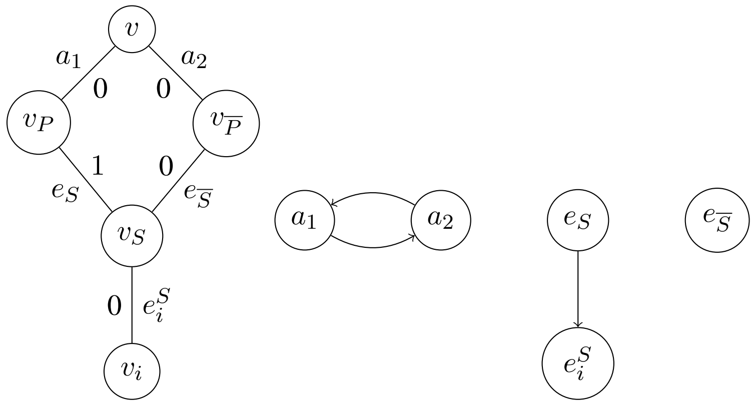


Figure 2: Illustration of the **SAT** reduction with $\Delta(G) = 3$. Intermediate triangles are related to clauses or variables.



(a) Graph $G$.　　(b) Digraph $D$.

Figure 3: Illustration of the Set Cover Problem reduction.

terized by the cost of the tree. The results hold even if the dependency relations occur only between adjacent edges, which seems a natural feature in practical applications.

Given a set $U$ and a family of subsets $\mathcal{C} \subseteq 2^U$, the Set Cover Problem (SCP) consists in finding a subfamily $\mathcal{S} \subseteq \mathcal{C}$ of minimum cardinality such that $\bigcup_{S \in \mathcal{S}} S = U$. Unless P = NP, it is inapproximable to $(1 - \Omega(1)) \ln |U|$, even when $|\mathcal{C}|$ is polynomial in $|U|$ (Dinur e Steurer [2014]). Moreover, it is $W[2]$-hard parameterized by the cardinality of the solution (Downey e Fellows [1999]).

We present a reduction from this problem to $\mathtt{DCMST}(G, D, w)$, as illustrated in Figure 3, that preserves the solution values. Starting with vertices $v$, $v_P$, $v_{\overline{P}}$ and edges $a_1 = \{v, v_P\}$, $a_2 = \{v, v_{\overline{P}}\}$, $G$ also includes: for each $S \in \mathcal{C}$, vertex $v_S$; for each $i \in U$, vertex $v_i$; for each $S \in \mathcal{C}$, edges $e_S = \{v_S, v_P\}$ and $e_{\overline{S}} = \{v_S, v_{\overline{P}}\}$; and edge $e_i^S = \{v_S, v_i\}$, if $i \in S, i \in U, S \in \mathcal{C}$. In $D$, $a_1$ and $a_2$ are mutually dependent, $e_S$ is a dependency for $e_i^S, i \in S, S \in \mathcal{C}$; all other vertices are isolated. $w$ is defined as $w_{e_S} = 1, \forall S \in \mathcal{C}$; $w_e = 0$, for any other edge $e$.

One component of $D$ is a directed cycle of length 2, and the other ones are directed stars and isolated vertices. Note also that $G$ has $|U|$ leaves and is bipartite, with one partition formed by $v_P, v_{\overline{P}}$ and $v_i, i \in U$. Once again, $D$ satisfies $\Delta^-(D) = 1$.

**Theorem 3.1.** *$\mathtt{DCMST}(G, D, w)$ is APX-hard, not being approximable to $(1 - \Omega(1)) \ln |V(G)|$ unless P = NP. Moreover, it is $W[2] -$ hard parameterized by the cost of the solution tree. The results hold even if G is bipartite, the dependency relations occur only between adjcent edges of G, and D has diameter 1 (composed by 2-cycles and stars).*

*Proof.* We prove that the reduction maps an instance $(U, \mathcal{C})$ of SCP into an instance $(G, D, w)$ of DCMST, while keeping the optimum value. Let $\mathcal{S} \subseteq \mathcal{C}$ be a feasible solution of SCP. We build a feasible solution $T$ to $\mathtt{DCMST}(G, D, w)$ as follows: edges $a_1$ and $a_2$ are in $T$, since they depend on each other; for each $S \in \mathcal{C}$, choose edge $e_S$ or $e_{\overline{S}}$, which have no dependency, depending on whether $S \in \mathcal{S}$ or $S \notin \mathcal{S}$, respectively; since $\mathcal{S}$ covers $U$, each $v_i$ is incident to an edge with dependencies satisfied, so choose that edge, making all the $v_i$'s leaves in $T$. Note that $T$ is a tree and, since the only edges with nonzero weight are the $e_S$'s, $\sum_{e \in E(T)} w_e = |S|$.

Let $T$ be a feasible solution for $\texttt{DCMST}(G, D, w)$ with weight $\sum_{e \in E(T)} w_e$. Due to their dependencies, $a_1$ and $a_2$ are in $T$. Because of that, and since $e_S$ and $e_{\overline{S}}$ are a cut in $G$, exactly one of them is in $T$, for each $S \in \mathcal{C}$. We build a solution $\mathcal{S}$ for the $\texttt{SCP}$ with $\mathcal{S} = \{S \in \mathcal{C} : e_S \in T\}$. Because $e_S$ is dependency for all $e_i^S$ and no $v_i$ can be isolated, $\mathcal{S}$ covers $U$, thus $\mathcal{S}$ is feasible. Trivially, $|S| = \sum_{e \in E(T)} w_e$. □

## 4. Polynomial cases

We present two cases where $\texttt{DCMST}(G, D, w)$ is solvable in polynomial time. In both of them, the strategy is to decompose the problem into a polynomial number of $\texttt{MST}$ subproblems. This aim is achieved thanks to the specific structure of $D$.

**Theorem 4.1.** *If $D$ has $O(\log_2(n(G)))$ components, and each of them is either an oriented cycle or an arborescence whose subjacent graph is a star, $\texttt{DCMST}(G, D, w)$ can be solved in polynomial time.*

*Proof.* If $D$ is a directed cycle, then $\texttt{DCMST}(G, D, w)$ is infeasible unless $G$ is itself a tree. When $D$ is an arborescence whose subjacent graph is a star, we delete its root $r$ and contract the incident edge (of $G$), obtaining $G'$ (we do this again in case of $D \setminus r$ is connected, obtaining $G''$) and solve $\texttt{MST}$ for the resulting graph $G'$ ($G''$).

When $D$ is a union of oriented cycles and directed stars, we have a choice to make for each of its (say $k$) components: for each directed cycle, we decide to include none or all of its edges in the solution; for each directed star, we decide to include or not the root of $D$ (and its unique descendant node, if the root is a leaf) and then allow or forbid all the other vertices of the star (edges of $G$) to take part of the solution. This way, we have $2^k$ $\texttt{MST}$ subproblems to consider, each of them solvable in polynomial time. If $k = O(\log_2(n(G)))$, then all these subproblems can be solved in polynomial time. □

**Theorem 4.2.** *$\texttt{DCMST}(G, D, w)$ can be solved in polynomial time, if $D$ is an arborescence whose subjacent graph is a caterpillar.*

*Proof.* Let $D(P \cup L, A)$ be an arborescence, where $P$ induces the main path and $L$ comprimes the leaves of the subjacent caterpillar. First, suppose that $D$ is rooted at $p_1 \in P$. Then, $p_1$ has exactly one or two neighbors in $P$:

- In the first case, consider $P = \{p_1, p_2, \ldots, p_k\}$. Note that there is an arc from $p_i$ to $p_{i+1}$, $1 \leq i \leq k - 1$. Let $L_i \subseteq L$ be the set of leaves adjacent to $p_i$, $1 \leq i \leq k$. Observe that $\texttt{DCMST}(G, D, w)$ can be decomposed into $k$ subproblems. The $i$th subproblem is the Minimum Spanning Tree Problem where the edges from $\{p_1, p_2, \ldots, p_i\}$ *must* be chosen and the edges from $L_1 \cup \cdots \cup L_i$ *can* be chosen (it may be infeasible if the first set induces a cycle in $G$). Since each of these subproblems can be solved in polynomial time, and there are $k$ of them, $\texttt{DCMST}(G, D, w)$ can be solved in polynomial time with its optimal solution being the one whose cost is minimum among the optimal solutions of the feasible subproblems.

- If $p_1$ has two neighbors in $P$, consider $P = \{p_1, p_1^1, p_2^1, \ldots, p_k^1, p_1^2, p_2^2, \ldots, p_l^2\}$, $k + l = |P| - 1$, such that $p_1, p_1^1, p_2^1, \ldots, p_k^1$ and $p_1, p_1^2, p_2^2, \ldots, p_l^2$ are directed paths in $D$. Also, consider $L_1 \subseteq L$ as the set of leaves incident to $p_1$, and let $L_i^1, L_j^2 \subseteq L$ be the analogously defined sets for $p_i^1$ and $p_j^2$, respectively, $1 \leq i \leq k$, $1 \leq j \leq l$. Note that $\texttt{DCMST}(G, D, w)$ can be decomposed into $kl$ subproblems. We index the subproblems with tuples $(i, j)$, $1 \leq i \leq k$, $1 \leq j \leq l$. Subproblem $(i, j)$ is the Minimum Spanning Tree Problem in the subgraph of $G$ induced by $p_1$, $p_a^1$, $p_b^2$, $L_a^1$ and $L_b^2$ edges, $1 \leq a \leq i$, $1 \leq b \leq j$. Moreover, the tree must contain edges $p_1, p_a^1$ and $p_b^2$, for $1 \leq a \leq i$, $1 \leq b \leq j$, and so the problem may be infeasible if these edges induce a cycle. Argumenting similarly as in the first case, $\texttt{DCMST}(G, D, w)$ can be solved in polynomial time with its optimal solution being the one whose cost is minimum among the optimal solutions of the feasible subproblems.

To finish the proof, we consider the case where $D$ is rooted at an $L$ vertex. It is clear that the corresponding edge must be part of any feasible solution for $\texttt{DCMST}(G, D, w)$ . So we contract it and fall back into the previous cases. □

## 5. ILP formulations

There are several Integer Linear Programming formulations for MST, based on different characterizations of spanning trees. To model $\texttt{L-DCMST}(G, D, w)$ and $\texttt{A-DCMST}(G, D, w)$, it suffices to extend these formulations. We show here formulations DCUT and MTZ for MST, which are the basis for our computational experiments.

### 5.1. DCUT

Given $G = (V, E)$ and $w : E \to \mathbb{R}^+$, we select a vertex $r \in V$ for the role of root. A spanning tree in $G$ is associated with an arborescence (rooted at $r$) in the digraph obtained from $G$ by replacing each edge by two arcs with opposite directions. So, we define variables $x_{uv}, y_{uv}, y_{vu}, \forall\{u, v\} \in E$. $x_{uv} = 1$ means that edge $\{u, v\}$ is in the solution (which implies that either $y_{uv} = 1$ or $y_{vu} = 1$). By $\delta^+(S)$, we denote the set of edges with exactly one endpoint in $S$, and by $N(v)$, we denote the set of neighbors of $v$. The model then follows, where connectivity is ensured by the so-called directed cut (DCUT) constraints (4).

$$
\begin{aligned}
\min \quad & \sum_{uv \in E} w_{uv} x_{uv} && && (1)\\
\text{s.t} \quad & x_{uv} = y_{uv} + y_{vu}, && \forall uv \in E && (2)\\
& \sum_{uv \in E} (y_{uv} + y_{vu}) = n(G) - 1 && && (3)\\
& \sum_{uv \in \delta^+(S)} y_{uv} \geq 1, && \forall S \subset V : r \in S && (4)\\
& \sum_{u \in N(v)} y_{uv} = 1, && \forall v \in V \setminus \{r\} && (5)\\
& \sum_{u \in N(r)} y_{ur} = 0 && && (6)\\
& y \in \mathbb{B}^{2m(G)} && && (7)\\
& x \in \mathbb{B}^{m(G)} && && (8)
\end{aligned}
$$

Although redundant in the formulation, constraints (5)-(6) are added for better computational performance.

### 5.2. MTZ

Given $G = (V, E)$ and $w : E \to \mathbb{R}^+$, we select a vertex $r \in V$ for the role of root. Again, we define variables $x_{uv}, y_{uv}, y_{vu}, \forall\{u, v\} \in E$. Moreover, we use variables $l_v \in \mathbb{R}, \forall v \in V$ to define a label for each vertex. An arc $(u, v)$ is chosen only if the label of $v$ is greater than the label of $u$. This strategy avoids cycles and can be modeled by the Muller-Tucker-Zemli (MTZ) inequalities.

The resulting formulation is presented below, where $N(v)$ denotes the set of neighbors of $v$, and $d(u, v)$ stands for the distance between vertices $u$ and $v$ in $G$.

$$
\begin{aligned}
\min \quad & \sum_{uv \in E} w_{uv} x_{uv} && && (9)\\
\text{s.a} \quad & x_{uv} = y_{uv} + y_{vu}, && \forall uv \in E && (10)\\
& \sum_{uv \in E} (y_{uv} + y_{vu}) = n(G) - 1 && && (11)\\
& \sum_{u \in N(v)} y_{uv} = 1, && \forall v \in V \setminus \{r\} && (12)\\
& \sum_{u \in N(r)} y_{ur} = 0 && && (13)\\
& l_r = 1 && && (14)\\
& l_u - l_v + 1 \leq (n(G) - d(r,v))(1 - y_{uv}), && \forall uv \in E, v \neq r && (15)\\
& l_v - l_u + 1 \leq (n(G) - d(r,u))(1 - y_{vu}), && \forall uv \in E, v \neq r && (16)\\
& 1 + d(r,v) \leq l_v \leq n(G), && \forall v \in V && (17)\\
& l \in \mathbb{R}^{n(G)} && && (18)\\
& y \in \mathbb{B}^{2m(G)} && && (19)\\
& x \in \mathbb{B}^{m(G)} && && (20)
\end{aligned}
$$

We observe that Constraints (11) are implied by (12)-(13). However, their inclusion leads to better computational results.

## 5.3. Dependency constraints

Let $N^{-}(e)$ denote the in-neighborhood of $e \in E$ in $D$. Problems L-DCMST$(G, D, w)$ and A-DCMST$(G, D, w)$ can be modeled by adding constraints (21) and (22), respectively, to a spanning tree integer programming formulation. We embeded them in formulations DCUT and MTZ.

$$
\begin{aligned}
& \sum_{e_1 \in N^{-}(e_2)} x_{e_1} \geq x_{e_2}, && \forall e_2 \in E : N^{-}(e_2) \neq \emptyset && (21)\\
& x_{e_1} \geq x_{e_2}, && \forall e_2 \in E : N^{-}(e_2) \neq \emptyset, \forall e_1 \in N^{-}(e_2) && (22)
\end{aligned}
$$

## 6. Computational experiments

To test for both L-DCMST$(G, D, w)$ and A-DCMST$(G, D, w)$, we designed a set of 90 instances where $D$ is an arborescence. Each instance $(G, D, w)$ is generated from a tuple $(n, d, b)$, where $n = |V(G)|$, $d$ is the density of $G$, and $b$ is the branchinhg factor of $D$, that is, each nonleaf vertex of $D$ has outdegree at most $b$. We have 2 instances for each $n \in \{30, 60, 90, 120, 150\}$, $d \in \{0.25, 0.50, 0.75\}$ and $b = \{2, 4, 8\}$.

The following tables present the running times of DCUT and MTZ on the standard CPLEX B&B. The root $r$ is chosen to be the vertex in the input graph whose label is minimum. In order to properly compare DCUT and MTZ, we ran MTZ both in serial and parallel executions, since DCUT could not be ran in parallel, due to limitations of CPLEX when using callbacks. * means the instance has unknown optimum, ** means execution has been aborted due to insufficient memory and *** means execution surpassed a time limit of 500,000 seconds and then has been aborted. Considering the average running time, we chose MTZ for performing further tests.

| $n$ | $d$ | $b$ | $i$ | Optimum | MTZ (s) | MTZ (Serial) (s) | DCUT (s) |
|---|---|---|---|---|---|---|---|
| 30 | 0.25 | 2 | 0 | 138 | 0.01 | 0.01 | 0.03 |
| 30 | 0.25 | 2 | 1 | 94 | 0.01 | 0.01 | 0.02 |
| 30 | 0.25 | 4 | 0 | 96 | 0.02 | 0.02 | 0.02 |
| 30 | 0.25 | 4 | 1 | 82 | 0.02 | 0.02 | 0.02 |
| 30 | 0.25 | 8 | 0 | 102 | 0.01 | 0.01 | 0.01 |
| 30 | 0.25 | 8 | 1 | 87 | 0.01 | 0.01 | 0.03 |
| 30 | 0.50 | 2 | 0 | 142 | 0.01 | 0.01 | 0.02 |
| 30 | 0.50 | 2 | 1 | 123 | 0.11 | 0.19 | 0.29 |
| 30 | 0.50 | 4 | 0 | 88 | 0.02 | 0.03 | 0.05 |
| 30 | 0.50 | 4 | 1 | 80 | 0.02 | 0.01 | 0.03 |
| 30 | 0.50 | 8 | 0 | 77 | 0.02 | 0.01 | 0.03 |
| 30 | 0.50 | 8 | 1 | 63 | 0.01 | 0.01 | 0.01 |
| 30 | 0.75 | 2 | 0 | 119 | 0.09 | 0.25 | 0.59 |
| 30 | 0.75 | 2 | 1 | 115 | 0.22 | 0.41 | 0.59 |
| 30 | 0.75 | 4 | 0 | 70 | 0.03 | 0.03 | 0.02 |
| 30 | 0.75 | 4 | 1 | 90 | 0.09 | 0.20 | 0.22 |
| 30 | 0.75 | 8 | 0 | 69 | 0.02 | 0.02 | 0.04 |
| 30 | 0.75 | 8 | 1 | 60 | 0.01 | 0.01 | 0.02 |
| 60 | 0.25 | 2 | 0 | 202 | 0.48 | 0.47 | 0.82 |
| 60 | 0.25 | 2 | 1 | 237 | 0.59 | 0.78 | 1.66 |
| 60 | 0.25 | 4 | 0 | 163 | 0.14 | 0.09 | 0.11 |
| 60 | 0.25 | 4 | 1 | 166 | 0.06 | 0.06 | 0.09 |
| 60 | 0.25 | 8 | 0 | 154 | 0.04 | 0.04 | 0.08 |
| 60 | 0.25 | 8 | 1 | 156 | 0.10 | 0.09 | 0.07 |
| 60 | 0.50 | 2 | 0 | 226 | 1.48 | 3.26 | 5.17 |
| 60 | 0.50 | 2 | 1 | 228 | 2.27 | 3.77 | 7.60 |
| 60 | 0.50 | 4 | 0 | 154 | 0.23 | 0.21 | 0.33 |
| 60 | 0.50 | 4 | 1 | 168 | 0.89 | 1.06 | 1.68 |
| 60 | 0.50 | 8 | 0 | 123 | 0.21 | 0.39 | 0.47 |
| 60 | 0.50 | 8 | 1 | 164 | 0.69 | 0.81 | 1.20 |
| 60 | 0.75 | 2 | 0 | 189 | 1.97 | 2.37 | 4.75 |
| 60 | 0.75 | 2 | 1 | 203 | 1.68 | 2.52 | 8.49 |
| 60 | 0.75 | 4 | 0 | 144 | 1.12 | 1.10 | 2.58 |
| 60 | 0.75 | 4 | 1 | 141 | 0.27 | 0.40 | 0.64 |
| 60 | 0.75 | 8 | 0 | 121 | 0.50 | 0.67 | 1.05 |
| 60 | 0.75 | 8 | 1 | 124 | 0.26 | 0.24 | 0.48 |
| 90 | 0.25 | 2 | 0 | 342 | 3.61 | 8.82 | 10.02 |
| 90 | 0.25 | 2 | 1 | 349 | 10.87 | 30.45 | 104.16 |
| 90 | 0.25 | 4 | 0 | 251 | 1.59 | 2.31 | 2.79 |
| 90 | 0.25 | 4 | 1 | 243 | 1.03 | 0.63 | 0.89 |
| 90 | 0.25 | 8 | 0 | 198 | 0.12 | 0.11 | 0.15 |
| 90 | 0.25 | 8 | 1 | 210 | 0.42 | 0.37 | 0.68 |
| 90 | 0.50 | 2 | 0 | 338 | 216.35 | 478.86 | 256.85 |
| 90 | 0.50 | 2 | 1 | 320 | 36.50 | 333.65 | 158.36 |
| 90 | 0.50 | 4 | 0 | 237 | 3.72 | 6.19 | 7.78 |
| 90 | 0.50 | 4 | 1 | 222 | 4.85 | 7.17 | 6.64 |
| 90 | 0.50 | 8 | 0 | 201 | 1.14 | 1.31 | 1.73 |
| 90 | 0.50 | 8 | 1 | 193 | 1.61 | 2.17 | 3.83 |
| 90 | 0.75 | 2 | 0 | 316 | 107.51 | 227.79 | 584.20 |
| 90 | 0.75 | 2 | 1 | 320 | 39.34 | 408.48 | 208.99 |
| 90 | 0.75 | 4 | 0 | 212 | 21.78 | 44.03 | 50.27 |
| 90 | 0.75 | 4 | 1 | 218 | 20.04 | 84.50 | 43.03 |
| 90 | 0.75 | 8 | 0 | 183 | 2.94 | 2.87 | 3.41 |
| 90 | 0.75 | 8 | 1 | 169 | 3.68 | 2.78 | 5.46 |
|  |  |  |  | Average | 9.08 | 30.77 | 27.56 |

Table 1: Running times of integer formulations for instances with at most 90 vertices.

| $n$ | $d$ | $b$ | $i$ | Optimum | `MTZ` (s) | `MTZ` (Serial) (s) | `DCUT` (s) |
|---|---|---|---|---|---|---|---|
| 120 | 0.25 | 2 | 0 | 434 | 683.87 | 1476.26 | 2359.79 |
| 120 | 0.25 | 2 | 1 | 466 | 600.08 | 3317.28 | 8020.24 |
| 120 | 0.25 | 4 | 0 | 300 | 2.06 | 1.73 | 4.48 |
| 120 | 0.25 | 4 | 1 | 321 | 4.85 | 5.94 | 4.11 |
| 120 | 0.25 | 8 | 0 | 233 | 0.60 | 0.50 | 1.00 |
| 120 | 0.25 | 8 | 1 | 246 | 0.76 | 0.63 | 0.93 |
| 120 | 0.50 | 2 | 0 | 398 | 3489.66 | 6613.45 | 6802.91 |
| 120 | 0.50 | 2 | 1 | 420 | 2923.59 | 2470.64 | 8719.21 |
| 120 | 0.50 | 4 | 0 | 302 | 141.83 | 194.11 | 570.71 |
| 120 | 0.50 | 4 | 1 | 295 | 11.86 | 13.50 | 12.99 |
| 120 | 0.50 | 8 | 0 | 231 | 3.66 | 2.97 | 4.97 |
| 120 | 0.50 | 8 | 1 | 254 | 10.55 | 12.82 | 17.27 |
| 120 | 0.75 | 2 | 0 | 403 | 4960.16 | 5090.58 | 16322.59 |
| 120 | 0.75 | 2 | 1 | 409 | 1958.91 | 3899.00 | 6163.77 |
| 120 | 0.75 | 4 | 0 | 289 | 982.21 | 1391.13 | 2379.59 |
| 120 | 0.75 | 4 | 1 | 300 | 206.16 | 863.92 | 292.57 |
| 120 | 0.75 | 8 | 0 | 212 | 4.61 | 5.85 | 4.85 |
| 120 | 0.75 | 8 | 1 | 241 | 19.77 | 13.60 | 32.83 |
| 150 | 0.25 | 2 | 0 | 538 | 1305.78 | 1677.46 | 6352.16 |
| 150 | 0.25 | 2 | 1 | 536 | 2263.90 | 6219.27 | 42304.22 |
| 150 | 0.25 | 4 | 0 | 355 | 9.33 | 14.10 | 11.27 |
| 150 | 0.25 | 4 | 1 | 383 | 7.20 | 6.75 | 7.76 |
| 150 | 0.25 | 8 | 0 | 321 | 11.69 | 14.01 | 9.44 |
| 150 | 0.25 | 8 | 1 | 321 | 2.93 | 2.50 | 2.56 |
| 150 | 0.50 | 2 | 0 | 495 | 4286.03 | 6250.46 | 10614.80 |
| 150 | 0.50 | 2 | 1 | 546 | 63243.97 | 111057.72 | 412665.05 |
| 150 | 0.50 | 4 | 0 | 362 | 2462.97 | 12326.15 | 5108.39 |
| 150 | 0.50 | 4 | 1 | 359 | 196.81 | 213.07 | 165.97 |
| 150 | 0.50 | 8 | 0 | 294 | 44.18 | 54.66 | 55.28 |
| 150 | 0.50 | 8 | 1 | 309 | 62.46 | 65.28 | 47.83 |
| 150 | 0.75 | 2 | 0 | 483 | 32807.35 | 96188.49 | 84107.06 |
| 150 | 0.75 | 2 | 1 | * | ** | ** | *** |
| 150 | 0.75 | 4 | 0 | 363 | 4308.08 | 8943.04 | 6955.49 |
| 150 | 0.75 | 4 | 1 | 376 | 11197.05 | 7757.59 | 32463.48 |
| 150 | 0.75 | 8 | 0 | 273 | 41.45 | 77.68 | 79.91 |
| 150 | 0.75 | 8 | 1 | 273 | 49.55 | 56.74 | 105.44 |
| | | | | Average | 2207.70 | 4860.03 | 7061.93 |

Table 2: Running times of integer formulations for instances with more than 90 vertices.

As an attempt to improve the CPLEX performance with the MTZ formulation, we customized its branch-and-bound in several points. In the following, we describe the two modifications that produced the best results.

First, we propose a method for finding an initial solution based on iteratively solving the problem restricted to a spanning subgraph of $G$. To define these iterations, let $f : E \to \mathbb{R}^+$ be defined as $f(e) = w_e + \min\{f(e') : e' \in N^-(e)\}$. In other words, $f(e)$ is the weight of a shortest path in $D$ from a source to $e$ (while considering the weights given by $w$). Sort $E' = E \setminus S$, where $S$ is the source set of $D$, according to the non-decreasing order defined by $f$, to obtain $E' = \{e_1, e_2, \ldots, e_{m(G)-|S|}\}$. For $k = 1, 2, \ldots, \log(n(G))$, let $P_k$ be the problem restricted to $G_k = (V, S \cup E'_k)$ and $D_k = (S \cup E'_k, A(S \cup E'_k))$, where $E'_k$ are the first $k\frac{m(G)-|S|}{\log(n(G))}$ edges of $E'$. We solve $P_k$, $k \geq 3$, until we find a feasible subproblem. We use its optimal solution as an initial feasible solution.

Besides providing this solution to CPLEX, we also prioritizing the branching of some $x$

variables. Such variables are related to edges whose distance to the source of $D$ is a third of the height of $D$.

The next table presents the computational results obtained by these two modifications on instances with CPLEX B&B running time above 100 seconds. In comparison with the B&B of CPLEX, we obtained an average improvement of 29.69%

| $n$ | $d$ | $b$ | $i$ | CPLEX B&B (s) | Technique (s) | Improvement (%) |
|---|---|---|---|---|---|---|
| 90 | 0.50 | 2 | 0 | 216.35 | 31.17 | 85.59 |
| 90 | 0.75 | 2 | 0 | 107.51 | 91.10 | 15.26 |
| 120 | 0.25 | 2 | 0 | 683.87 | 703.03 | -2.80 |
| 120 | 0.25 | 2 | 1 | 600.08 | 353.40 | 41.10 |
| 120 | 0.50 | 2 | 0 | 3489.66 | 3191.55 | 8.54 |
| 120 | 0.50 | 2 | 1 | 2923.59 | 1422.61 | 51.34 |
| 120 | 0.50 | 4 | 0 | 141.83 | 84.82 | 40.19 |
| 120 | 0.75 | 2 | 0 | 4960.16 | 2610.95 | 47.36 |
| 120 | 0.75 | 2 | 1 | 1958.91 | 792.18 | 59.56 |
| 120 | 0.75 | 4 | 0 | 982.21 | 209.14 | 78.70 |
| 120 | 0.75 | 4 | 1 | 206.16 | 171.69 | 16.72 |
| 150 | 0.25 | 2 | 0 | 1305.78 | 173.18 | 86.73 |
| 150 | 0.25 | 2 | 1 | 2263.90 | 2606.37 | -15.12 |
| 150 | 0.50 | 2 | 0 | 4286.03 | 6470.05 | -50.95 |
| 150 | 0.50 | 4 | 0 | 2462.97 | 2465.26 | -0.09 |
| 150 | 0.50 | 4 | 1 | 196.81 | 82.58 | 58.04 |
| 150 | 0.75 | 2 | 0 | 32807.35 | 42337.39 | -29.04 |
| 150 | 0.75 | 4 | 0 | 4308.08 | 2621.25 | 39.15 |
| 150 | 0.75 | 4 | 1 | 11197.05 | 7406.94 | 33.84 |
| Average | | | | 3952.54 | 3885.50 | 29.69 |
| Median | | | | 1958.91 | 792.18 | 39,15 |

## References

Darmann, A., Pferschy, U., Schauer, J., e Woeginger, G. J. (2011). Paths, trees and matchings under disjunctive constraints. *Discrete Applied Mathematics*, 159(16):1726 – 1735.

Dinur, I. e Steurer, D. (2014). Analytical approach to parallel repetition. In *Proc. of ACM Symposium on Theory of Computing*, STOC'14, p. 624–633.

Downey, R. G. e Fellows, M. R. (1999). *Parameterized Complexity*. Springer.

Gary, M. R. e Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-completeness*.

Samer, P. e Urrutia, S. (2015). A branch and cut algorithm for minimum spanning trees under conflict constraints. *Optimization Letters*, 9(1):41–55.

Viana, L. A. (2016). árvore geradora com dependências mínima. Master's thesis, Federal University of Ceará.

Zhang, R., Kabadi, S., e Punnen, A. (2011). The minimum spanning tree problem with conflict constraints and its variations. *Discrete Optimization*, 8(2):191 – 205.