Minimum Activation Cost Node-Disjoint Paths in Graphs with Bounded Treewidth

Hasna Mohsen Alqahtani and Thomas Erlebach

Department of Computer Science, University of Leicester, Leicester, UK.
{hmha1|t.erlebach}@leicester.ac.uk

Abstract. In *activation network* problems we are given a directed or undirected graph $G = (V, E)$ with a family $\{f_{uv} : (u, v) \in E\}$ of monotone non-decreasing activation functions from D^2 to $\{0, 1\}$, where D is a constant-size subset of the non-negative real numbers, and the goal is to find activation values x_v for all $v \in V$ of minimum total cost $\sum_{v \in V} x_v$ such that the activated set of edges satisfies some connectivity requirements. We propose algorithms that optimally solve the *minimum activation cost of k node-disjoint st-paths* (*st*-MANDP) problem in $O(tw((5 + tw)|D|)^{2tw+2}|V|^3)$ time and the *minimum activation cost of node-disjoint paths* (MANDP) problem for k disjoint terminal pairs $(s_1, t_1), \ldots, (s_k, t_k)$ in $O(tw((4 + 3tw)|D|)^{2tw+2}|V|)$ time for graphs with treewidth bounded by tw.

1 Introduction

In *activation network* problems, we are given an *activation network*, which is a directed or undirected graph $G = (V, E)$ together with a family $\{f_{uv} : (u, v) \in E\}$ of monotone non-decreasing activation functions from D^2 to $\{0, 1\}$, where D is a constant-size subset of the non-negative real numbers. The activation of an edge depends on the chosen values from the domain D at its endpoints. An edge $(u, v) \in E$ is *activated* for chosen values x_u and x_v if $f_{uv}(x_u, x_v) = 1$. An activation function f_{uv} for $(u, v) \in E$ is called *monotone non-decreasing* if $f_{uv}(x_u, x_v) = 1$ implies $f_{uv}(y_u, y_v) = 1$ for any $y_u \geq x_u$, $y_v \geq x_v$. The goal is to determine activation values $x_v \in D$ for all $v \in V$ so that the total activation cost $\sum_{v \in V} x_v$ is minimized and the activated set of edges satisfies some connectivity requirements. As activation network problems are computationally difficult in arbitrary graphs, it is meaningful to investigate whether restricted graph classes admit efficient algorithms. In this paper we consider the problems of finding *minimum activation cost k node-disjoint st-paths* (*st*-MANDP) and finding *minimum activation cost node-disjoint paths* (MANDP) between k disjoint terminals pairs, $(s_1, t_1), \ldots, (s_k, t_k)$, for graphs of bounded treewidth. Throughout this paper we consider undirected simple graphs.

The problem of finding k node/edge-disjoint paths between two nodes s and t in a given graph can be solved in polynomial time using network flow techniques [1], but no polynomial-time algorithm is known for the network activation setting. The problem of finding node/edge-disjoint paths (NDP/EDP) between

k terminals pairs, $(s_1, t_1), \ldots, (s_k, t_k)$, in a given graph is NP-complete if k is part of the input [5] (and already for $k = 2$ in directed graphs [4]). However, the problem is polynomial-time solvable for undirected graphs when k is fixed [11]. Scheffler [12] gave a linear-time algorithm that follows a classical bottom-up approach to solve the NDP problem for arbitrary k in graphs of bounded treewidth. In this paper we adapt Scheffler's algorithm [12] to the *activation network* version of node-disjoint path problems and devise polynomial-time optimal algorithms for solving the MANDP problem and the st-MANDP problem in graphs of bounded treewidth.

Related work. Activation network problems were introduced recently by Panigrahi [10]. The problem of finding the *minimum activation st-path* (MAP) can be solved optimally in polynomial-time [10]. However, the *minimum activation edge-disjoint st-paths* (st-MAEDP) problem is NP-hard [10]. The *minimum spanning activation tree* (MSpAT) problem is NP-hard to approximate within a factor of $o(\log n)$. The MSpAT problem is a special case of the problems of finding the *minimum Steiner activation network* (MSAN) and the *minimum edge/node-connected activation network* (MEAN/MNAN) (activating a network with k edge/node-disjoint paths between every pair of vertices). Therefore, these problems are also NP-hard to approximate within $o(\log n)$. As mentioned in [10], there exist $O(\log n)$-approximation algorithms for MSpAT, and for MEAN and MNAN in the case of $k = 2$. There is a 2-approximation algorithm for the st-MANDP problem and a $2k$-approximation algorithm for the st-MAEDP problem [8]. The st-MAEDP and st-MANDP problems when $k = 2$ have been studied recently by the authors [2]. They show that a ρ-approximation algorithm for the *minimum activation 2 node-disjoint st-paths* (st-MA2NDP) problem implies a ρ-approximation algorithm for the *minimum activation 2 edge-disjoint st-paths* (st-MA2EDP) problem. They also obtained a 1.5-approximation algorithm for the st-MA2NDP problem and hence for the st-MA2EDP problem. Furthermore, they showed that the st-MANDP problem for the restricted version of activation networks with $|D| = 2$ and a single activation function for all edges can be solved in polynomial time for arbitrary k (except for one case of the activation function, in which they require $k = 2$). However, the st-MAEDP problem remains NP-hard under this restriction [10, 2]. It is not yet known whether the st-MANDP and st-MAEDP problems for an arbitrary constant-size D and fixed $k \geq 2$ are NP-hard.

Activation network problems can be viewed as a generalization of several known problems in wireless network design such as power optimization problems. In power optimization problems, we are given a graph $G = (V, E)$ and each edge $(u, v) \in E$ has a threshold power requirement θ_{uv}. In the undirected case we say the edge (u, v) is activated for chosen values x_u and x_v if each of these values is at least θ_{uv}, and in the directed case it is activated if $x_u \geq \theta_{uv}$. [7] shows that a simple reduction to the shortest st-path problem can solve the *minimum power st-path* (MPP) problem in polynomial time for directed/undirected networks. The problem of finding *minimum power k node-disjoint st-paths* (st-MPkNDP) in directed graphs can be solved in polynomial time [6, 13]. However, the *mini-*

mum power k edge-disjoint st-paths (st-MPkEDP) problem is unlikely to admit even a polylogarithmic approximation algorithm for both the directed and undirected variants [6]. In power optimization, the MSpAT problem is APX-hard and the MEAN and MNAN problems have 4-approximation and 11/3-approximation algorithms, respectively. See [2, 8–10] for further motivation and applications of *activation network* problems.

Our results. We propose algorithms that optimally solve the st-MANDP problem in $O(tw((5+tw)|D|)^{2tw+2}|V|^3)$ time and the MANDP in $O(tw\ (4+3tw)^{2tw+2}\ |D|^{2tw+2}\ |V|)$ time for graphs with treewidth bounded by tw.

This paper is organized as follows. In Section 2, we introduce some notations and definitions used throughout this paper. In Section 3, we propose an exact algorithm that solves the st-MANDP problem on graphs with bounded treewidth tw in polynomial-time. Section 4 presents an optimal algorithm that solves the MANDP problem on graphs with bounded treewidth in linear-time.

2 Preliminaries

The class of graphs of bounded treewidth [11] has attracted attention due to the fact that many NP-complete problems for arbitrary graphs are polynomial or even linear time solvable in graphs of bounded treewidth.

Definition 1. *A tree-decomposition for a given graph $G = (V, E)$ is a pair $(\mathcal{X}, \mathcal{T})$ of a tree $\mathcal{T} = (I, F)$ and a family $\{X_i\}_{i \in I}$ of subsets of V (called bags) satisfying the following two conditions: (1) For every edge $(v, w) \in E$, there exists an $i \in I$ with $v \in X_i$ and $w \in X_i$. (2) For every vertex $v \in V$, the nodes $i \in I$ with $v \in X_i$ form a subtree of $\mathcal{T}$. The width of a tree-decomposition is $\max_{i \in I} |X_i| - 1$. The treewidth tw of the graph G is the minimum width among all possible tree-decompositions of the graph.*

As shown in [3], there exists a linear-time algorithm that checks whether a given graph $G = (V, E)$ has treewidth at most tw, for fixed tw, and outputs the tree-decomposition $(\mathcal{X}, \mathcal{T})$ of G.

Definition 2. *A tree-decomposition $(\mathcal{X}, \mathcal{T})$ is called a* nice *tree-decomposition, if $\mathcal{T}$ is a binary tree rooted at some $r \in I$ that satisfies the following:*

- *Each node is either a leaf, or has exactly one or two children.*
- *Let $i \in I$ be a leaf. Then $X_i \subseteq \{u : (u, v) \in E\} \cup \{v\} = N[v]$ for some $v \in V$.*
- *For every edge $(u, v) \in E$, there is a leaf $i \in I$ such that $\{u, v\} \subseteq X_i$.*
- *Let $j \in I$ be the only child of $i \in I$, then either $X_i = X_j \cup \{v\}$ or $X_i = X_j \setminus \{v\}$. The node i is called an introduce node or forget node, respectively.*
- *Let $j, j' \in I$ be the two child nodes of a node $i \in I$, then $X_j = X_{j'} = X_i$. The node i is called a join node of $\mathcal{T}$.*

Scheffler [12] shows that any given tree-decomposition $(\mathcal{X}, \mathcal{T})$ can be easily converted into a nice tree-decomposition.

Theorem 1 ([12]). *A nice tree-decomposition of width tw and size $O(|V(G)|)$ can be constructed for a graph $G = (V, E)$ with treewidth at most tw in linear time, if tw is a fixed constant.*

To give a simpler description of our algorithms, we assume that every bag of a leaf of the nice tree-decomposition consists of two vertices that are connected by an edge in G, and that for every edge (u, v) of G there is exactly one leaf $i \in I$ such that $u, v \in X_i$ (and we say that the edge (u, v) is *associated* with that leaf $i \in I$). A nice tree-decomposition with this property can also be easily obtained from any given tree-decomposition in linear time. In the remainder of the paper, we assume that the input graph G is given as a simple undirected graph together with a nice tree-decomposition of width at most tw.

Let us define X_i^+ to be the set of all vertices in X_j for all nodes $j \in I$ such that $j = i$ or j is a descendant of i. Let G_i^+ describe the partial graphs of G. For a leaf node i, G_i^+ is the subgraph of G with vertex set X_i and the edge of G that is associated with i. For a non-leaf node i, G_i^+ is the graph that is the union of G_j^+ over all children j of i. Note that the graph G_r^+ for the root r of the tree-decomposition is equal to G.

3 Minimum Activation Cost k Node-Disjoint st-Paths

Let $G = (V, E)$ be an activation network and $s, t \in V$ be a pair of source and destination vertices. The goal of the st-MANDP problem is to find activation values $\{x_v : v \in V\}$ of minimum total cost $\sum_{v \in V} x_v$ such that the activated set of edges contains k node disjoint st-paths $\mathcal{P}^{st} = P_1, \ldots, P_k$. In this section we present a polynomial-time algorithm that solves the st-MANDP problem optimally in the case of graphs of bounded treewidth. The algorithm follows a bottom-up approach based on a nice tree-decomposition. In our algorithm, each node i of the tree-decomposition has a table tab_i to store its computed information. The algorithm computes a number of possible sub-solutions per tree node $i \in I$ based on the information computed previously for its children. Let $\mathcal{P} = P_1, \ldots, P_k$ be a solution for the st-MANDP problem. Let $\mathcal{P}^i = \mathcal{P}\ [G_i^+]$ be the induced solution in a partial graph G_i^+ (i.e., the set of vertices and edges that are both in $\mathcal{P}$ and in G_i^+). The interaction between $\mathcal{P}^i$ in G_i^+ and the rest of the graph happens only in vertices of X_i. The partial solution $\mathcal{P}^i$ can therefore be represented by an activation-value function $\Lambda_i : X_i \to D$ and a state function $\beta_i : X_i \to \{s, t, o, \infty\} \cup \{0, 1, \ldots, k\} \cup (X_i \times \{c\})$. As the algorithm needs to consider partial solutions in G_i^+ that cannot necessarily be completed to form a global solution, we call any subgraph of G_i^+ (with suitable activation values) a *partial solution* (or *sub-solution*) if it contains all vertices in $\{s, t\} \cap X_i^+$ and each connected component C satisfies one of the following conditions:

1. C is an isolated vertex $v \in X_i$.
2. C is a simple path having both end-vertices $v, w \in X_i$ and containing neither s nor t.

3. C contains s but not t and consists of several (at least one) paths that are node-disjoint (apart from meeting in s), each connecting s to a vertex in X_i (or the same condition with s and t exchanged).
4. C contains s and t and consists of some number of paths from s to t, some paths from s to a vertex in X_i, and some paths from t to a vertex in X_i. All these paths are node-disjoint (apart from having s or t in common). If $s \notin X_i$, s has degree k, otherwise s has degree $\leq k$, and likewise for t.

Intuitively, partial solutions are subgraphs of G_i^+ that could potentially be completed to a global solution if the rest of the graph is of suitable form.

State Function. A partial solution $\mathcal{P}^i$ in G_i^+ can be represented by a state function $\beta_i : X_i \to \{s, t, o, \infty\} \cup \{0, 1, \ldots, k\} \cup (X_i \times \{c\})$ as follows:

- For any $v \in X_i$, we set $\beta_i(v) = 0$ iff $v \in \mathcal{P}^i$ and has degree zero in $\mathcal{P}^i$, i.e., v is a connected component of $\mathcal{P}^i$ that satisfies condition 1 above.
- For any $v \in X_i$, we set $\beta_i(v) = \infty$ iff $v \notin \mathcal{P}^i$.
- For any $v \in X_i \setminus \{s, t\}$, we set $\beta_i(v) = o$ iff v has degree 2 in $\mathcal{P}^i$.
- For any $v \in X_i \cap \{s, t\}$, we set $\beta_i(v) = k' \in \{1, \ldots, k\}$ iff a connected component C of $\mathcal{P}^i$ contains v together with k' incident edges, i.e., C satisfies condition 3 or 4 above. We call v an occupied vertex if $k' = k$.
- For any $v \in X_i \setminus \{s, t\}$, we set $\beta_i(v) = u \in \{s, t\}$ iff there is a non-empty path $u, \ldots, v$ not containing s or t as internal node that is a subgraph of a connected component of $\mathcal{P}^i$ that satisfies condition 3 or 4 above and v has degree 1 in that component.
- For any pair $u, v \in X_i \setminus \{s, t\}$, we set $\beta_i(u) = (v, c)$ and $\beta_i(v) = (u, c)$ iff u and v are connected with a path $u, \ldots, v$ that is a maximal connected component of $\mathcal{P}^i$ that does not contain s and t, i.e., the maximal connected component $u, \ldots, v$ satisfies condition 2 above.

3.1 Processing the Tree Decomposition

Let $val(\beta_i, \Lambda_i)$ denote the optimal cost of an assignment of activation values for G_i^+ which satisfies the restriction Λ_i and activates a partial solution satisfying β_i. In each row of the table tab_i of tree node $i \in I$, we store a solution of a unique combination of a state function β_i and a function of activation values Λ_i (each vertex of X_i has a state value and is assigned an activation value). Additionally, we store the activation cost value $val(\beta_i, \Lambda_i)$ for the solution. We can compute the sub-solution tables in a bottom-up approach.

Leaf. Let $i \in I$ be a leaf, $X_i = \{u, v\}$. Let (β_i, Λ_i) be any row of tab_i. We distinguish the following cases and define $val(\beta_i, \Lambda_i)$ for each case. Each case corresponds to a possible sub-solution in G_i^+. Recall that G_i^+ is a single edge. The sub-solution's cost $val(\beta_i, \Lambda_i)$ is set to $\sum_{v \in X_i} \Lambda_i(v)$ if one of the following cases applies:

- $\beta_i(w) \in \{0, \infty\}$ for all $w \in X_i$, and $\beta_i(w) = 0$ for $w \in X_i \cap \{s, t\}$. Intuitively, this means that the sub-solution has no edges.

- $\beta_i(u) = v$ and $\beta_i(v) = 1$ and $u \notin \{s,t\}$ and $v \in \{s,t\}$ and $f_{uv}(\Lambda_i(u), \Lambda_i(v)) = 1$. Intuitively, the sub-solution is a path with one edge and one endpoint equal to s or t. (The roles of u and v can be exchanged.)
- $\beta_i(u) = \beta_i(v) = 1$ and $u, v \in \{s,t\}$ and $f_{uv}(\Lambda_i(u), \Lambda_i(v)) = 1$. Intuitively, the sub-solution is a path with one edge containing s and t.
- $\beta_i(u) = (v,c)$ and $\beta_i(v) = (u,c)$ and $u, v \notin \{s,t\}$ and $f_{uv}(\Lambda_i(u), \Lambda_i(v)) = 1$. Intuitively, the sub-solution is a path with one edge not containing s or t.

If none of the above cases applies, $val(\beta_i, \Lambda_i) = +\infty$. In these cases, (β_i, Λ_i) does not represent a subgraph of G_i^+ that could be part of a global solution.

Introduce. Let $i \in I$ be an introduce node, and $j \in I$ its only child. We have $X_j \subset X_i$, $|X_i| = |X_j| + 1$ and let v be the additional isolated vertex in X_i. The vertex v always has state value $\beta_i(v) \in \{0, \infty\}$. For every row (β_j, Λ_j) in tab_j, there are $2|D|$ rows in tab_i such that $\beta_i(u) = \beta_j(u)$ and $\Lambda_i(u) = \Lambda_j(u)$ for all $u \in X_i \setminus \{v\}$. The sub-solution cost $val(\beta_i, \Lambda_i)$ for these rows is set to $val(\beta_j, \Lambda_j) + \Lambda_i(v)$ if $v \notin \{s,t\}$ or if $v \in \{s,t\}$ and $\beta_i(v) = 0$. Otherwise $val(\beta_i, \Lambda_i) = +\infty$.

Forget. Let $i \in I$ be a forget node, and $j \in I$ its only child. We have $X_i \subset X_j$, $|X_j| = |X_i| + 1$ and let v be the discarded vertex. For each (β_i, Λ_i), consider all (β_j, Λ_j) that agree with (β_i, Λ_i) for all $u \in X_i$ and satisfy that $\beta_j(v) = k$ if $v \in \{s,t\}$ and $\beta_j(v) \in \{o, \infty\}$ otherwise. The sub-solution's cost $val(\beta_i, \Lambda_i)$ is the minimum of $val(\beta_j, \Lambda_j)$ over all these (β_j, Λ_j).

Join. Let $i \in I$ be a join node, and $j, j' \in I$ its two children. We have $X_i = X_j = X_{j'}$. Let (β_j, Λ_j) and $(\beta_{j'}, \Lambda_{j'})$ be rows of tab_j and $tab_{j'}$, respectively. When we combine these solutions, their connected components may get merged into larger components, and we need to ensure that we do not create any cycles that do not contain s and t. For this purpose, we construct an auxiliary graph H_i with vertex set X_i and edge set $E_{H_i} = \{uv | \beta_j(v) = (u,c)\} \cup \{uv \mid \beta_{j'}(v) = (u,c)\}$ to help us detect cases where such a cycle would be created. The algorithm combines the sub-solutions (β_j, Λ_j) and $(\beta_{j'}, \Lambda_{j'})$ if both have the same activation-value function ($\Lambda_j(u) = \Lambda_{j'}(u)$ for all $u \in X_i$) and the union does not satisfy any of the following conditions (where the roles of j and j' can be exchanged):

C_1. H_i contains a cycle (which may also consist of just one pair of parallel edges).
C_2. There is a vertex $v \in X_i$ with $\beta_j(v) = o$ and $\beta_{j'}(v) \neq 0$.
C_3. There is a vertex $v \in X_i$ with $\beta_j(v) = \infty$ and $\beta_{j'}(v) \neq \infty$.
C_4. There is a vertex $v \in X_i$ with $\beta_j(v) = \beta_{j'}(v) \in \{s,t\}$.
C_5. There is a vertex $v \in X_i \cap \{s,t\}$ with $\beta_j(v) = k'$ and $\beta_{j'}(v) = k''$ and $k' + k'' \notin \{0, 1, \ldots, k\}$.

We compute the state function β_i of the combination of β_j and $\beta_{j'}$. Consider any $v \in X_i$:

- If $\beta_j(v) = \beta_{j'}(v) = \sigma \in \{0, \infty\}$, then $\beta_i(v) = \sigma$.
- If $\beta_j(v) = \sigma \in \{o, s, t, k\}$ and $\beta_{j'}(v) = 0$, then $\beta_i(v) = \sigma$.
- If $\beta_j(v) = s$ and $\beta_{j'}(v) = t$, then $\beta_i(v) = o$.
- If $v \in \{s,t\}$ and $\beta_j(v) = k'$ and $\beta_{j'}(v) = k''$ for any $k' + k'' \in \{0, 1, \ldots, k\}$, then $\beta_i(v) = k' + k''$.

– If $\beta_j(v) = (x,c)$ and $\beta_{j'}(v) = (y,c)$ for any $x,y \in X_i$, then $\beta_i(v) = o$.
– If $\beta_j(v) = 0$ and $\beta_{j'}(v) = (x,c)$ and the maximal path in H_i that starts at v ends at u with $\beta_j(u) = 0$ and $\beta_{j'}(u) = (y,c)$ (or vice versa, i.e., the path ends at u with $\beta_j(u) = (y,c)$ and $\beta_{j'}(u) = 0$) for any $x,y \in X_i$, then $\beta_i(v) = (u,c)$ and $\beta_i(u) = (v,c)$.
– If $\beta_j(v) = s$ and $\beta_{j'}(v) = (x,c)$ and the maximal path in H_i that starts at v ends at u with $\beta_j(u) = (y,c)$ and $\beta_{j'}(u) = t$ (or vice versa) for any $x,y \in X_i$, then $\beta_i(v) = \beta_i(u) = o$.
– If $\beta_j(v) = \sigma \in \{s,t\}$ and $\beta_{j'}(v) = (x,c)$ and the maximal path in H_i that starts at v ends at u with $\beta_j(u) = (y,c)$ and $\beta_{j'}(u) = 0$ (or vice versa) for any $x,y \in X_i$, then $\beta_i(v) = o$ and $\beta_i(u) = \sigma$.

The value of the combined solution $val(\beta_i, \Lambda_i)$ is calculated as the minimum summation value over all pairs of sub-solutions that can be combined to produce (β_i, Λ_i) minus the activation cost of X_i.

Extracting the solution at the root. The algorithm checks all the solutions (β_r, Λ_r) of the root bag X_r such that vertices in $\{s,t\} \cap X_r$ are occupied and all other vertices have state value o or ∞. In this case (β_r, Λ_r) is a feasible solution. The output of the algorithm is the solution of minimum cost value among all the feasible solutions obtained at the root. For each row (β_i, Λ_i) of bag X_i, we store the rows of X_i's children that were used in the calculation of $val(\beta_i, \Lambda_i)$. Computing the optimum solution is possible by traversing top-down in the decomposition tree to the leaves (traceback).

3.2 Analysis

Let an instance of the problem be given by an activation network $G = (V,E)$ with bounded treewidth tw and terminals $s,t \in V$. Let $\mathcal{P}^{OPT}$ represent an optimal solution for this instance. We use $C(\mathcal{P}^i)$ to denote the activation cost of the induced solution $\mathcal{P}^i$ of $\mathcal{P}$ in a partial graph G_i^+.

Lemma 1. *The algorithm runs in $O(tw((5+tw)|D|)^{2tw+2}|V|^3)$ time.*

Proof. The running-time of the algorithm depends on the size of the tables and the combination of tables during the bottom-up traversal. Each vertex $w \in X_i$ has $5 + (|X_i| - 1)$ possible state values from $\{0,s,t,o,\infty\} \cup ((X_i \setminus \{w\}) \times \{c\})$ if $w \notin \{s,t\}$ and $k+1$ possible state values if $w \in X_i \cap \{s,t\}$. If $|X_i \cap \{s,t\}| \leq 1$, the table tab_i in a processed bag X_i contains no more than $O(k(5+tw)^{tw+1}|D|^{tw+1})$ rows corresponding to the possible state functions and the $|D|$ possible activation values for each vertex of X_i. Assume that $\{s,t\} \subseteq X_i$ and $\beta_i(s) = k_s$, $\beta_i(t) = k_t$ and $c_q = \{v \in X_i : \beta_i(v) = q\}$ such that $k_s, k_t \in \{0,\ldots,k\}$ and $q \in \{s,t\}$. Since every path in the partial solution starting at s leads to t or to a vertex in X_i then $k_s - c_s = k_t - c_t$. Hence, there is at most one k_t when leaving the other values fixed. Therefore, the table tab_i in a processed bag X_i contains no more than $O(k(5+tw)^{tw+1}|D|^{tw+1})$ rows. Considering all possible row combinations for two tables for a join node and noting that $k \leq |V|$, we see that the computation of the state functions needs $O(tw)$ time for each combination and $O(tw((5+tw)|D|)^{2tw+2}|V|^3)$ time overall. $\square$

Lemma 2. *For any processed bag X_i, let $\mathcal{P}_i^{OPT}$ be the induced solution of $\mathcal{P}^{OPT}$ in G_i^+ and $(\beta_i^{OPT}, \Lambda_i^{OPT})$ be the corresponding state function and activation values, then $val(\beta_i^{OPT}, \Lambda_i^{OPT}) \leq C(\mathcal{P}_i^{OPT})$.*

Lemma 3. *For any processed bag X_i, any solution (β_i, Λ_i) where $val(\beta_i, \Lambda_i) = c_i < \infty$ corresponds to a partial solution $\mathcal{P}^i$ with the following properties:*

- *The state function of $\mathcal{P}^i$ in X_i is β_i.*
- *The activation values of $\mathcal{P}^i$ in X_i are Λ_i.*
- *The total activation cost in X_i^+ is c_i.*
- *$\mathcal{P}^i$ contains the terminal s if $s \in X_i^+$, and the terminal t if $t \in X_i^+$.*

Due to space restrictions, the proofs of Lemma 2 and Lemma 3 (which use induction over the tree-decomposition) are deferred to the full version.

We say that (β_i, Λ_i) is a feasible solution at the root node i iff (β_i, Λ_i) is a partial solution with $\beta_i(v) = k$ for $v \in \{s,t\} \cap X_i$ and $\beta_i(v) \in \{o, \infty\}$ for all $v \in X_i \setminus \{s,t\}$. That means the partial solution in G_i^+ consists of k node-disjoint paths between s and t. The algorithm outputs the solution of minimum activation cost among all feasible solutions obtained at the root. From the above lemmas and the extracting part in the algorithm, we get the following theorem.

Theorem 2. *The st-MANDP problem can be solved optimally in $O(tw\ (5+tw)^{2tw+2}\ |D|^{2tw+2}\ |V|^3)$ time for graphs with treewidth bounded by tw.*

This algorithm can be used for the variant of the *st*-MANDP problem in graphs of bounded treewidth where s and t are assigned specified activation values $d \in D$ and $d' \in D$, respectively, by setting $\Lambda(s) = d$ and $\Lambda(t) = d'$. We recall the following theorem from [2]:

Theorem 3 ([2]). *If there is a ρ-approximation algorithm for the st-MA2NDP problem where s and t are assigned specified activation values, then there is a ρ-approximation algorithm for the st-MA2EDP problem.*

Corollary 1. *There exists a polynomial-time algorithm that optimally solves the st-MA2EDP problem for graphs of bounded treewidth.*

4 Minimum Activation Cost Node-Disjoint Paths between *k* Pairs of Terminals

Consider an instance of the problem given by an activation network $G = (V, E)$ and k disjoint pairs of terminals $(s_1, t_1), \ldots, (s_k, t_k)$. The goal of MANDP is to find activation values $\{x_v : v \in V\}$ of minimum total cost $\sum_{v \in V} x_v$ such that the activated set of edges contains k node disjoint paths $\mathcal{P} = P_1, \ldots, P_k$ in G where P_a is a path connecting s_a and t_a for all $a \in \{1, \ldots, k\}$. Define $\mathcal{S} = \{s_a | 1 \leq$

$a \leq k\} \cup \{t_a | 1 \leq a \leq k\}$. Let $\mathcal{N}_a = (s_a, t_a)$, for $1 \leq a \leq k$. In this section we modify the linear-time algorithm proposed in [12] that solves the NDP problem on graphs of bounded treewidth to solve the problem in the activation network case. We assume that k is arbitrary. As for the *st*-MANDP algorithm in Section 3, the algorithm stores all computed sub-solutions for each node i of the nice tree-decomposition in a table tab_i. For every node i, we define $\mathcal{O}_i = \{\mathcal{N}_a | s_a \in X_i^+$ and $t_a \notin X_i^+$ or $s_a \notin X_i^+$ and $t_a \in X_i^+\}$ and $\mathcal{Q}_i = \{\mathcal{N}_a | s_a \in X_i^+$ and $t_a \in X_i^+\}$. Let $\mathcal{P}^i$ be any subgraph of the partial graph G_i^+. If $\mathcal{P}^i$ is a partial solution, the algorithm characterizes $\mathcal{P}^i$ by an activation function $\Lambda_i : X_i \to D$ and a state function $\beta_i : X_i \to \{0, 1, \infty\} \cup \mathcal{O}_i \cup \mathcal{Q}_i \cup (X_i \times \{c, d\})$. Here, we say that $\mathcal{P}^i$ is a partial solution if it contains all $v \in X_i^+ \cap \mathcal{S}$ and any connected component C of $\mathcal{P}^i$ satisfies one of the following conditions:

1. C is a simple path P_a connecting s_a and t_a (and not containing any other vertex in $\mathcal{S}$).
2. C is an isolated vertex $v \in X_i$.
3. C is a simple path having both end-vertices $v, w \in X_i$ and not containing any vertices of $\mathcal{S}$.
4. C is a simple path that connects a source $s_a \in X_i^+$ with a vertex $s'_a \in X_i$. If $\mathcal{N}_a \in \mathcal{Q}_i$, there is another (possibly empty) path C' that connects the terminal $t_a \in X_i^+$ with a vertex $t'_a \in X_i$. (The roles of s_a and t_a can be exchanged.)

State Function. The algorithm characterizes $\mathcal{P}^i$ by an activation-value function $\Lambda_i : X_i \to D$ and a state function $\beta_i : X_i \to \{0, 1, \infty\} \cup \mathcal{O}_i \cup \mathcal{Q}_i \cup (X_i \times \{c, d\})$. Given a partial solution $\mathcal{P}^i$, the state function β_i is defined as follows:

– For any $v \in X_i$, we set $\beta_i(v) = 0$ iff $v \in \mathcal{P}^i$ and v has degree zero in $\mathcal{P}^i$.
– For any $v \in X_i$, we set $\beta_i(v) = \infty$ iff $v \notin \mathcal{P}^i$.
– For any $v \in X_i$, we set $\beta_i(v) = 1$ iff v is either an inner vertex of a path in $\mathcal{P}^i$ or $v \in \mathcal{S}$ and $\mathcal{P}^i$ contains v together with an incident edge. We call v an occupied vertex.
– For any $v \in X_i$, we set $\beta_i(v) = \mathcal{N}_a$ iff there is a path $s_a, \ldots, v$ (or a path $t_a, \ldots, v$) in $\mathcal{P}^i$, v has degree 1 in $\mathcal{P}_i$, and either $\mathcal{N}_a \in \mathcal{O}_i$ or we have that $\mathcal{N}_a \in \mathcal{Q}_i$ and t_a (or s_a) is in X_i has degree 0 in $\mathcal{P}_i$.
– For any pair $u, v \in X_i$, we set $\beta_i(u) = (v, d)$ and $\beta_i(v) = (u, d)$ iff there are two paths $s_a, \ldots, u$ and $t_a, \ldots, v$ in $\mathcal{P}^i$ for any $\mathcal{N}_a \in \mathcal{Q}_i$ and u, v have degree 1. We say that s_a and t_a are a disconnected pair.
– For any pair $u, v \in X_i$, we set $\beta_i(u) = (v, c)$ and $\beta_i(v) = (u, c)$ iff u and v are connected with a path $u, \ldots, v$ that is a maximal connected component of $\mathcal{P}^i$ that does not contain any vertices from $\mathcal{S}$ and u, v have degree 1.

4.1 Processing the Tree Decomposition

As for the *st*-MANDP algorithm in Section 3, each tab_i of X_i stores multiple rows and each row represents a unique combination of a state function β_i and an activation function Λ_i (each vertex of X_i has a state and is assigned an activation value). The cost value $val(\beta_i, \Lambda_i)$ of the represented sub-solution is also stored in tab_i.

Leaf. Let $i \in I$ be a leaf, $X_i = \{u, v\}$ and $(u, v) \in E_i$, where E_i is the set of edges associated with i. We distinguish the following cases and define $val(\beta_i, \Lambda_i)$ for each case. Let (β_i, Λ_i) be any row of tab_i. If none of the following cases applies, $val(\beta_i, \Lambda_i) = +\infty$. The sub-solution's cost is $val(\beta_i, \Lambda_i) = \sum_{v \in X_i} \Lambda_i(v)$ if one of the following cases applies:

- $\beta_i(w) \in \{0, \infty\}$ for all $w \in X_i$ and $\beta_i(w) = 0$ for $w \in \mathcal{S}$. Intuitively, the sub-solution has no edges.
- $\beta_i(u) = \beta_i(v) = 1$ and $u, v \in \mathcal{N}_a$ for any $\mathcal{N}_a \in \mathcal{Q}_i$ and $f_{uv}(\Lambda_i(u), \Lambda_i(v)) = 1$. Intuitively, the sub-solution is a path with one edge containing s_a and t_a for some $\mathcal{N}_a \in \mathcal{Q}_i$.
- $\beta_i(u) = \mathcal{N}_a$, $\beta_i(v) = 1$ and $u \notin \mathcal{S}$ and $v \in \mathcal{N}_a$ for any $\mathcal{N}_a \in \mathcal{O}_i$ and $f_{uv}(\Lambda_i(u)$ $, \Lambda_i(v)) = 1$. Intuitively, the sub-solution is a path with one edge and one endpoint equal to s_a or t_a for any $\mathcal{N}_a \in \mathcal{O}_i$. (The roles of u and v can be exchanged.)
- $\beta_i(u) = (v, c)$, $\beta_i(v) = (u, c)$ and $u, v \notin \mathcal{S}$ and $f_{uv}(\Lambda_i(u), \Lambda_i(v)) = 1$. Intuitively, the sub-solution is a path with one edge not containing any vertices of $\mathcal{S}$.

Introduce and *Forget* nodes are processed in a similar way to Section 3.1.

Join. Let $i \in I$ be a join node, and $j, j' \in I$ its two children. We have $X_i = X_j = X_{j'}$. Let (β_j, Λ_j) and $(\beta_{j'}, \Lambda_{j'})$ be rows of tab_j and $tab_{j'}$, respectively. We consider an auxiliary graph H_i with vertex set X_i and edge set $E_{H_i} = \{uv | \beta_j(v) = (u, c)\} \cup \{uv \mid \beta_{j'}(v) = (u, c)\}$ to help computing the state function β_i of the combination of β_j and $\beta_{j'}$. The algorithm combines the sub-solutions (β_j, Λ_j) and $(\beta_{j'}, \Lambda_{j'})$ if both have the same activation-value function ($\Lambda_j(u) = \Lambda_{j'}(u)$ for all $u \in X_i$) and the union does not satisfy the following (the roles of j and j' could be exchanged):

C_1. H_i contains a cycle.
C_2. There is a vertex $v \in X_i$ with $\beta_j(v) = 1$ and $\beta_{j'}(v) \neq 0$.
C_3. There is a vertex $v \in X_i$ with $\beta_j(v) = \infty$ and $\beta_{j'}(v) \neq \infty$.
C_4. There is a vertex $v \in X_i$ with $\beta_j(v) = \mathcal{N}_a$ and $\beta_{j'}(v) = \mathcal{N}_b$ and $a \neq b$.
C_5. There is a vertex $v \in X_i$ such that $\beta_j(v) = \mathcal{N}_a$ and $\beta_{j'}(v) = (u, d)$ for any $u \in X_i$.
C_6. There is a vertex $v \in X_i$ with $\beta_j(v) = (u, d)$ and $\beta_{j'}(v) = (w, d)$ for any $u, w \in X_i$.

We compute the state function β_i of the combination of β_j and $\beta_{j'}$. Consider $v \in X_i$:

- If $\beta_j(v) = \beta_{j'}(v) = \sigma \in \{0, \infty\}$, then $\beta_i(v) = \sigma$.
- If $\beta_j(v) = 1$ and $\beta_{j'}(v) = 0$, then $\beta_i(v) = 1$.

- If $\beta_j(v) = \mathcal{N}_a$ and $\beta_{j'}(v) = 0$, we distinguish two cases: If there is another vertex $u \in X_i$ with $\beta_j(u) = 0$ and $\beta_{j'}(u) = \mathcal{N}_a$, then $\beta_i(u) = (v, d)$ and $\beta_i(v) = (u, d)$. Otherwise, $\beta_i(v) = \mathcal{N}_a$.
- If $\beta_j(v) = \mathcal{N}_a$ and $\beta_{j'}(v) = \mathcal{N}_a$, then $\beta_i(v) = 1$.
- If $\beta_j(v) = (x, c)$ and $\beta_{j'}(v) = (y, c)$ for any $x, y \in X_i$, then $\beta_i(v) = 1$.
- If $\beta_j(v) = (u, d)$ and $\beta_{j'}(v) = 0$ and $\beta_j(u) = (v, d)$ and $\beta_{j'}(u) = 0$ for any $u \in X_i$, then $\beta_i(u) = (v, d)$ and $\beta_i(v) = (u, d)$.
- If $\beta_j(v) = 0$ and $\beta_{j'}(v) = (x, c)$ and the maximal path in H_i that starts at v ends at u with $\beta_j(u) = 0$ and $\beta_{j'}(u) = (y, c)$ or $\beta_j(u) = (y, c)$ and $\beta_{j'}(u) = 0$ for any $x, y \in X_i$, then $\beta_i(v) = (u, c)$ and $\beta_i(u) = (v, c)$.
- If $\beta_j(v) = \mathcal{N}_a$ and $\beta_{j'}(v) = (x, c)$ and the maximal path in H_i that starts at v ends at u with $\beta_j(u) = (y, c)$ and $\beta_{j'}(u) = \mathcal{N}_a$ for any $x, y \in X_i$ and $a \in \{1, \ldots, k\}$, then $\beta_i(v) = \beta_i(u) = 1$.
- If $\beta_j(v) = (u, d)$ and $\beta_{j'}(v) = (x, c)$ and the maximal path in H_i that starts at v ends at u with $\beta_j(u) = (v, d)$ and $\beta_{j'}(u) = (y, c)$ for any $x, y \in X_i$, then $\beta_i(v) = \beta_i(u) = 1$.
- If $\beta_j(v) = \mathcal{N}_a$ and $\beta_{j'}(v) = (x, c)$ and the maximal path in H_i that starts at v ends at u with $\beta_j(u) = (y, c)$ and $\beta_{j'}(u) = 0$ for any $x, y \in X_i$ and $a \in \{1, \ldots, k\}$, then $\beta_i(v) = 1$ and $\beta_i(u) = \mathcal{N}_a$.
- If $\beta_j(v) = (u, d)$ and $\beta_{j'}(v) = (x, c)$ and the maximal path in H_i that starts at v ends at w with $\beta_j(w) = (y, c)$ and $\beta_{j'}(u) = 0$ for any $x, y \in X_i$, then $\beta_i(v) = 1$ and $\beta_i(u) = (w, d)$ and $\beta_i(w) = (u, d)$.
- If $\beta_j(v) = (u, d)$ and $\beta_{j'}(v) = (x, c)$ and $\beta_j(u) = (v, d)$ and $\beta_{j'}(u) = (y, c)$ the maximal path in H_i that starts at v ends at w with $\beta_j(w) = (z, c)$ and $\beta_{j'}(w) = 0$ and the maximal path in H_i that starts at u ends at q with $\beta_j(q) = (z', c)$ and $\beta_{j'}(q) = 0$ for any $x, y, z, z' \in X_i$, then $\beta_i(v) = \beta_i(u) = 1$, $\beta_i(q) = (w, d)$ and $\beta_i(w) = (q, d)$.

The value of the combined solution is the minimum summation value over all pairs of sub-solutions that can be combined to produce (β_i, Λ_i), minus the activation cost of X_i.

Extracting the solution at the root can be done similarly as in Section 3.1.

This completes the description of the MANDP algorithm. In this algorithm, each vertex $w \in X_i$ has $3 + 2(|X_i| - 1) + |X_i|$ possible state values from the set $\{0, 1, \infty\} \cup ((X_i \setminus \{w\}) \times \{c, d\}) \cup \{\mathcal{N}_a \mid \mathcal{N}_a \in \mathcal{O}_i\} \cup \{\mathcal{N}_a \mid \mathcal{N}_a \in \mathcal{Q}_i, \mathcal{N}_a \cap X_i \neq \emptyset\}$. (Note that the union of the latter two sets has cardinality at most $|X_i|$ if a feasible solution exists.) Then, the table tab_i contains no more than $O((4 + 3tw)^{tw+1}|D|^{tw+1})$ rows corresponding to the possible state functions and the $|D|$ possible activation values for each vertex of X_i. Each table has $O((4 + 3tw)^{tw+1}|D|^{tw+1})$ rows and the combination of a join node requires $O(tw(4 + 3tw)^{2tw+2}\ |D|^{2tw+2})$ time. The proofs of the running time and correctness of the MANDP algorithm are similar to the proofs of the running time and correctness of the *st*-MANDP algorithm. Due to space restrictions, the analysis is omitted from this paper, and we close this section with the following theorem.

Theorem 4. *The MANDP problem for graphs with bounded treewidth tw can be solved optimally in $O(tw(4 + 3tw)^{2tw+2}|D|^{2tw+2}|V|)$ time.*

5 Conclusion

We have presented a polynomial-time algorithm that optimally solves the *st*-MANDP problem for the case of graphs with bounded treewidth. We also showed that the linear-time algorithm for the NDP problem for graphs of bounded treewidth that has been presented in [12] can be modified to obtain a linear-time algorithm for the problem in activation networks. One open problem is to obtain a faster or even linear-time algorithm for the *st*-MANDP problem in graphs of bounded treewidth. It would also be interesting to investigate the *st*-MAEDP problem for graphs of bounded treewidth.

Acknowledgements We would like to thank the anonymous reviewers for their thorough and helpful comments on an earlier version of this paper. Their comments helped making the paper clearer and improving the analysis of the running time of the algorithm in Section 3.

References

1. Ahuja, R. K., Magnanti, T. L., Orlin, J. B.: Network flows: Theory, Algorithms and Applications. Prentice Hall, New Jersey (1993)
2. Alqahtani, H. M., Erlebach, T.: Approximation Algorithms for Disjoint *st*-Paths with Minimum Activation Cost. In: CIAC 2013, LNCS, vol. 7878, pp. 1–12. Springer, Heidelberg (2013)
3. Bodlaender, H. L.: A linear time algorithm for finding tree-decompositions of small treewidth. In: ACM STOC 1993, pp. 226–234. (1993)
4. Fortune, S., Hopcroft, J., Wyllie, J.: The directed subgraph homeomorphism problem. Theoretical Computer Science 10(2), 111–121 (1980)
5. Garey, M. R., Johnson, D. S.: Computers and Intractability. A Guide to the Theory of NP-Completeness. W. H. Freeman and Company, New York–San Francisco (1979)
6. Hajiaghayi, M. T., Kortsarz, G., Mirrokni, V. S., Nutov, Z.: Power optimization for connectivity problems. In: IPCO XI, LNCS, vol. 3509, pp. 349–361. Springer, Heidelberg (2005)
7. Lando, Y., Nutov, Z.: On minimum power connectivity problems. In: ESA 2007, LNCS, vol. 4698, pp. 87–98. Springer, Heidelberg (2007)
8. Nutov, Z.: Survivable network activation problems. In: LATIN 2012, LNCS, vol. 7256, pp. 594–605. Springer, Heidelberg (2012)
9. Nutov, Z.: Approximating Steiner networks with node-weights. SIAM J. Comput. 39(7), 3001–3022 (2010)
10. Panigrahi, D.: Survivable network design problems in wireless networks. In: 22nd Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 1014–1027. SIAM (2011)
11. Robertson, N., Seymour, P. D.: Graph Minors XIII. The Disjoint Paths Problem. In: J. Comb. Theory, Ser. B, vol. 63, pp. 65–110. (1995)
12. Scheffler, P. : A practical linear time algorithm for disjoint paths in graphs with bound tree-width. Technical Report 396, Dept. Mathematics, Technische Universität Berlin. (1994)
13. Srinivas, A., Modiano, E.: Finding Minimum Energy Disjoint Paths in Wireless Ad-Hoc Networks. Wireless Networks 11(4), 401–417 (2005)