Survivable Network Design for Group Connectivity in Low-Treewidth Graphs

Parinya Chalermsook* Syamantak Das† Guy Even‡ Bundit Laekhanukit§
Daniel Vaz¶

April 20, 2018

Abstract

In the *Group Steiner Tree* problem (GST), we are given a (vertex or edge)-weighted graph $G = (V, E)$ on n vertices, together with a root vertex r and a collection of groups $\{S_i\}_{i\in[h]} : S_i \subseteq V(G)$. The goal is to find a minimum-cost subgraph H that connects the root to every group. We consider a fault-tolerant variant of GST, which we call Restricted (Rooted) Group SNDP. In this setting, each group S_i has a demand $k_i \in [k], k \in \mathbb{N}$, and we wish to find a minimum-cost subgraph $H \subseteq G$ such that, for each group S_i, there is a vertex in the group that is connected to the root via k_i (vertex or edge) disjoint paths.

While GST admits $O(\log^2 n \log h)$ approximation, its higher connectivity variants are known to be *Label-Cover* hard, and for the vertex-weighted version, the hardness holds even when $k = 2$ (it is widely believed that there is no subpolynomial approximation for the Label-Cover problem [Bellare et al., STOC 1993]). More precisely, the problem admits no $2^{\log^{1-\epsilon} n}$-approximation unless NP $\subseteq$ DTIME($n^{\text{polylog(n)}}$). Previously, positive results were known only for the edge-weighted version when $k = 2$ [Gupta et al., SODA 2010; Khandekar et al., Theor. Comput. Sci., 2012] and for a relaxed variant where k_i disjoint paths from r may end at different vertices in a group [Chalermsook et al., SODA 2015], for which the authors gave a bicriteria approximation. For $k \geq 3$, there is no non-trivial approximation algorithm known for edge-weighted Restricted Group SNDP, except for the special case of the relaxed variant on trees (folklore).

Our main result is an $O(\log n \log h)$ approximation for Restricted Group SNDP that runs in time $n^{f(k,w)}$, where w is the treewidth of the input graph. This nearly matches the lower bound when k and w are constants. The key to achieving this result is a non-trivial extension of a framework introduced in [Chalermsook et al., SODA 2017]. This framework first embeds all feasible solutions to the problem into a dynamic program (DP) table. However, finding the optimal solution in the DP table remains intractable. We formulate a linear program relaxation for the DP and obtain an approximate solution via randomized rounding. This framework also

*Aalto University, Finland. **email:** chalermsook@gmail.com

†Indraprastha Institute of Information Technology Delhi, India. **email:** syamantak@iiitd.ac.in

‡Tel-Aviv University, Israel. **email**: guy@eng.tau.ac.il

§Max-Planck-Institut für Informatik, Germany & Shanghai University of Finance and Economics, China. **email**: blaekhan@mpi-inf.mpg.de

¶Max-Planck-Institut für Informatik, Germany & Graduate School of Computer Science, Saarland University, Germany. **email**: ramosvaz@mpi-inf.mpg.de

allows us to systematically construct DP tables for high-connectivity problems. As a result, we present new exact algorithms for several variants of survivable network design problems in low-treewidth graphs.

1 Introduction

Network design is an important subject in computer science and combinatorial optimization. The goal in network design is to build a network that meets some prescribed properties while minimizing the construction cost. *Survivable network design problems* (SNDP) are a class of problems where we wish to design a network that is resilient against link or node failures.

These problems have been phrased as optimization problems on graphs, where we are given an n-vertex (undirected or directed) graph $G = (V, E)$ with costs on edges or vertices together with a connectivity requirement $k : V \times V \to \mathbb{N}$. The goal is to find a minimum-cost subgraph $H \subseteq G$, such that every pair $u, v \in V$ of vertices are connected by $k(u, v)$ edge-disjoint (resp., openly vertex-disjoint) paths. In other words, we wish to design a network in which every pair of vertices remains connected (unless $k(u, v) = 0$), even after removing $k(u, v) - 1$ edges (or vertices). The *edge-connectivity* version of SNDP (EC-SNDP) models the existence of link failures and the *vertex-connectivity* (VC-SNDP) models the existence of both link and node failures. These two problems were known to be NP-hard and have received a lot of attention in the past decades (see, e.g., [26, 16, 32, 34]).

While VC-SNDP and EC-SNDP address the questions that arise from designing telecommunication networks, another direction of research focuses on the questions that arise from media broadcasting as in cable television or streaming services. In this case, we may wish to connect the global server to a single local server in each community, who will forward the stream to all the clients in the area through their own local network. The goal here is slightly different from the usual SNDP, as it is not required to construct a network that spans every client; instead, we simply need to choose a local server (or representative), which will take care of connecting to other clients in the same group. This scenario motivates the *Group Steiner Tree* problem (GST) and its fault-tolerant variant, the *Rooted Group SNDP*.

In Rooted Group SNDP, we are given a graph $G = (V, E)$ with costs on edges or vertices, a root vertex r, and a collection of subsets of vertices called *groups*, $S_1, \ldots, S_h$, together with connectivity demands $k_1, \ldots, k_h \in [k]$, $k \in \mathbb{N}$. The goal in this problem is to find a minimum cost subgraph $H \subseteq G$ such that H has k_i edge-disjoint (or openly vertex-disjoint) paths connecting the root vertex r to some vertex $v_i \in S_i$, for all $i \in [h]$. In other words, we wish to choose one representative from each group and find a subgraph of G such that each representative is k-edge-(or vertex)-connected to the root.

When $k = 1$, the problem becomes the well-known *Group Steiner Tree* (GST) problem. Here, we are given a graph $G = (V, E)$ with edge or vertex costs, a root r and a collection of subsets of vertices called groups, $S_1, \ldots, S_h \subseteq V$, and the goal is to find a minimum-cost subgraph $H \subseteq G$ that has a path to some vertex in each S_i, for $i \in [h]$. The GST problem is known to admit an $O(\log^3 n)$-approximation algorithm [22] and cannot be approximated to a factor of $\log^{2-\epsilon} n$ unless NP $\subseteq$ ZPTIME($n^{\text{polylog(n)}}$) [25].

The Rooted Group SNDP generalizes GST to handle fault tolerance. The case where $k = 2$ is studied in [27, 23], culminating in the $\tilde{O}(\log^4 n)$-approximation algorithm for the problem. For $k \geq 3$, there is no known non-trivial approximation algorithm. It is known among the experts that this problem is at least as hard as the *Label-Cover* problem[1].

Chalermsook, Grandoni and Laekhanukit [13] studied a relaxed version of the problem in which we are not restricted to connect to a single vertex in each group and thus need only k_i edge-disjoint paths connecting the root vertex to the whole group S_i. Despite being a relaxed condition, the problem remains as hard as the Label-Cover problem, and they only managed to design a bicriteria approximation algorithm.

To date, there is no known bicriteria or even sub-exponential-time poly-logarithmic approximation for Rooted Group SNDP when $k \geq 3$. The following is an intriguing open question:

> What are the settings (i.e., ranges of k or graph classes) in which Rooted Group SNDP admits a poly-logarithmic approximation?

In this paper, we focus on developing algorithmic techniques to approach the above question. We design poly-logarithmic algorithms for a special class of graphs – graphs with bounded treewidth – in the hope that it will shed some light towards solving the problem on a more general class of graphs, for instance, planar graphs (this is the case for the Steiner tree problem, where a sub-exponential-time algorithm for planar graphs is derived via decomposition into low-tree width instances [33]).

Our main technical building block is a dynamic program (DP) that solves rooted versions of EC-SNDP and VC-SNDP in bounded-treewidth graphs. However, a straightforward DP computation is not applicable for Restricted Rooted Group SNDP, simply because the problem is NP-hard on trees (so it is unlikely to admit a polynomial-size DP-table). Hence, we "embed" the DP table into a tree and apply a polylogarithmic approximation algorithm using randomized rounding of a suitable LP-formulation by Chalermsook et al. [12]. [2] We remark that when the cost is polynomially bounded (e.g., in the Word RAM model with words of size $O(\log n)$), polynomial-time algorithms for EC-SNDP and VC-SNDP follows from *Courcelle's Theorem* [17, 6] (albeit, with much larger running time). However, employing the theorem as a black-box does not allow us to design approximation algorithms for Restricted Rooted Group SNDP. ←D

To avoid confusion between the relaxed and restricted version of Rooted Group SNDP (usually having the same name in literature), we refer to our problem as Restricted Group SNDP. (For convenience, we also omit the word "rooted".)

1.1 Related Work

SNDP problems on restricted graph classes have also been studied extensively. When $k = 1$, the problems are relatively well understood. Approximation algorithms and PTAS have been developed for many graph classes: low-treewidth graphs [1, 18], metric-cost graphs [14], Euclidean graphs [9], planar graphs [8], and graphs of bounded genus [7]. [3] However, when $k \geq 2$, the complexity of ←D

[1]The hardness for the case of directed graph was shown in [27], but it is not hard to show the same result for

these problems remains wide open. Borradaile et al. [7, 10] showed an algorithm for $k = 1, 2, 3$ on planar graphs, but under the assumption that one can buy multiple copies of edges (which they called *relaxed connectivity* setting). Without allowing multiplicity, very little is known when $k \geq 2$: Czumaj et al. [19] showed a PTAS for $k = 2$ in unweighted planar graphs, and Berger et al. [3] showed an exact algorithm running in time $2^{O(w^2)}n$ for the uniform demand case (i.e., $k(u, v) = 2$ for all pairs (u, v)). Thus, without the relaxed assumption, with non-uniform demands or $k > 2$, the complexity of SNDP problems on bounded-treewidth graphs and planar graphs is not adequately understood.

The technique of formulating an LP from a DP table has been used in literature. It is known that any (discrete) DP can be formulated as an LP, which is integral [31]. (For Stochastic DP, please see, e.g., [30, 21, 11, 20].) However, the technique of producing a tree structure out of a DP table is quite rare. Prior to this paper the technique of rounding LP via a tree structure was used in [24] to approximate the Sparsest-Cut problem. The latter algorithm is very similar to us. However, while we embed a graph into a tree via a DP table, their algorithm works directly on the tree decomposition. We remark that our technique is based on the previous work in [12] with almost the same set of authors.

1.2 Hardness of Approximating Restricted Group SNDP

As mentioned, it is known among the experts that vertex-cost variant of the Restricted Group SNDP has a simple reduction for the *Label-Cover* problem and more generally, the *k-Constraint Satisfaction* problem (k-CSP). The original construction was given by Khandekar, Kortsarz and Nutov [27] for the Restricted Group SNDP on directed graphs. However, the same construction applies for the Vertex-Weighted Restricted Group SNDP. We are aware that this fact might not be clear for the readers. Thus, we provide the sketch of the proof in Appendix F.

We remark that the k-CSP hardness implies that even for $k_i \in \{0, 2\}$, Vertex-Weighted Restricted Group SNDP cannot be approximated to within a factor of $2^{\log^{1-\epsilon} n}$, for any $\epsilon > 0$, unless NP $\subseteq$ DTIME($n^{\text{polylog(n)}}$), and the approximation hardness is conjectured to be polynomial on n, say n^{δ} for some $0 < \delta < 1$, under the *Sliding Scale Conjecture* [2]. So far, we do not know of any non-trivial approximation for this problem for $k \geq 2$.

The edge-cost variant has been studied in [27, 23], and a polylogarithmic approximation is known for the case $k = 2$ [27]. For $k > 3$, there is no known non-trivial approximation algorithm. The relaxed variant where the k disjoint paths from the root may end at different vertices in each group S_i has also been studied in [27, 23]. Chalermsook, Grandoni and Laekhanukit proposed a bicriteria approximation algorithm for the Relaxed Restricted Group SNDP [13]; however, their technique is not applicable for the restricted version. Note that the hardness of the edge-cost variant of Relaxed Restricted Group SNDP is $k^{1/6-\epsilon}$, for any $\epsilon > 0$ [13]. It is not hard to construct the same hardness result for Restricted Group SNDP. We believe that Restricted Group SNDP is strictly harder than the relaxed variant.

undirected graphs.

[2] ⇒ Is this ok? –Daniel

[3] ⇒ Changed efficient to approximation –Daniel

1.3 Our Results & Techniques

Our main result is the following approximation result for Restricted Group SNDP.

Theorem 1.1. *There is a randomized algorithm that runs in time $n^{f(w,k)}$, for some function f, and returns, with high probability, a feasible solution to Restricted Group SNDP that has expected cost at the most $O(\log n \log h)$ times the cost of an optimal solution.*

←D

[4]

The proof of this theorem relies on the technique introduced in [12]. We give an overview of this technique and highlight how this paper departs from it.

In short, this technique "bridges" the ideas of dynamic program (DP) and randomized LP rounding in two steps[5]. Let us say that we would like to approximate optimization problem Π. In the first step, a "nice" DP table that captures the computation of the optimal solution is created, and there is a 1-to-1 correspondence between the DP solution and the solution to the problem. However, since the problem is NP-hard (in our case, even hard to approximate to within some poly-logarithmic factor), we could not follow the standard bottom-up computation of DP solutions. The idea of the second step is to instead write an LP relaxation that captures the computation of the optimal DP solution, and then use a randomized dependent rounding to get an approximate solution instead; the randomized rounding scheme is simply the well-known GKR rounding [22]. Roughly speaking, the size of the DP table is $n \cdot w^{O(w)}$, while the LP relaxation has $n^{O(w \log w)}$ variables and constraints, so we could get an $O(\log n \log h)$ approximation in time $n^{O(w \log w)}$.

The main technical hurdle that prevents us from using this technique to Restricted Group SNDP directly is that there was no systematic way to generate a "good" DP table for arbitrary connectivity demand k. (The previous result was already complicated even for $k = 1$.) This is where we need to depart from the previous work. We devise a new concept that allows us to systematically create such a DP table for any connectivity demand k. Our DP table has size $n \cdot f(k, w)$ for some function k and w, and it admits the same randomized rounding scheme in time $n^{g(k,w)}$, therefore yielding the main result.

As by-products, we obtain new DP algorithms for some well-studied variants of SNDP, whose running time depends on the treewidth of the input graph (in particular, $n \cdot f(k, w)$).

Subset connectivity problems: Subset k-Connectivity is a well-studied SNDP problem (Subset k-EC and Subset k-VC for edge and vertex connectivity, respectively). In this setting, all pairs of terminals have the same demands, i.e., $k(u, v) = k$ for all $u, v \in T$. This is a natural generalization of Steiner tree that has received attention [14, 32, 29].

Theorem 1.2. *There are exact algorithms for Subset k-EC and Subset k-VC that run in time $f_1(k, w)n$ for some function f_1. This result holds for vertex- or edge-costs.*

Rooted SNDP: Another setting that has been studied in the context of vertex connectivity requirements is the Rooted SNDP [15, 32]. In this problem, there is a designated terminal vertex

[4] ⇒ The success probability is 1, like in GKR; because after $O(\log n \log h)$ we can just cover the remaining groups one by one. –Daniel

[5] One may view our result as a "tree-embedding" type result. Please see [12] for more discussion along this line. Here we choose to present our result in the viewpoint of DP & LP.

$r \in T$, and all positive connectivity requirements are enforced only between r and other terminals, i.e. $k(u,v) > 0$ only if $u = r$ or $v = r$. For the edge connectivity setting, Rooted SNDP captures Subset k-EC[6].

Theorem 1.3. *There are exact algorithms for* Rooted EC-SNDP *and* Rooted VC-SNDP *that run in time $f_2(k,w)n$ for some function f_2. This result holds for costs on vertices or edges.*

Further technical overview: Let us illustrate how our approach is used to generate the DP table, amenable for randomized rounding. The following discussion assumes a certain familiarity with the notion of treewidth and DP algorithms in low-treewidth graphs.

Given graph $G = (V, E)$, let $\mathcal{T}$ be a tree decomposition of G having width w, i.e. each node $t \in V(\mathcal{T})$ corresponds to a *bag* $X_t \subseteq V(G) : |X_t| \leq w$. Let $\mathcal{T}_t$ denote the subtree of $\mathcal{T}$ rooted at t. For each $t \in V(\mathcal{T})$, let G_t denote the subgraph induced on all bags belonging to the subtree of $\mathcal{T}$ rooted at t, i.e. $G_t = G[\bigcup_{t \in \mathcal{T}_t} X_t]$. At a high level, DPs for minimization problems in low-treewidth graphs proceed as follows. For each node $t \in V(\mathcal{T})$, there is a profile $\pi_t \in \Pi$, and we define a DP cell $c[t, \pi_t]$ for each possible such profile, which stores the minimum-cost of a solution (a subgraph of G_t) that is consistent with the profile π_t. Then, a recursive rule is applied: Let t', t'' be the left and right children of t in $\mathcal{T}$ respectively. The DP makes a choice to "buy" a subset of edges $Y \subseteq E(G[X_t])$ (that appear in bag X_t) and derives the cost by minimizing over all profiles $\pi_{t'}, \pi_{t''}$ that are "consistent" with π_t:

$$c[t, \pi_t] = \min_{\pi_{t'}, \pi_{t''}, Y : (\pi_{t'}, \pi_{t''}, Y) \bowtie \pi_t} (\text{cost}(Y) + c[t', \pi_{t'}] + c[t'', \pi_{t''}])$$

where the sign $(\pi_{t'}, \pi_{t''}, Y) \bowtie \pi_t$ represents the notion of consistency between the profiles. Different optimization problems have different profiles and consistency rules. Often, consistency rules that are designed for connectivity-1 problems (such as Steiner tree) are not easily generalizable to higher connectivity problems (such as SNDP).

In this paper, we devise a new consistency rule (abbreviated by $\eqslantless$) for checking "reachability" (or connectivity 1) in a graph, which allows for easy generalization to handle high connectivity problems.

Roughly speaking, our consistency rule $\eqslantless$ solves the Steiner tree problem (connectivity-1 problem). To solve a connectivity-k problem, we have a DP cell $c[t, \vec{\pi}]$ for each $\vec{\pi} \in \Pi^k$. Then the consistency check is a "direct product" test for all coordinates, i.e.,

$$(\vec{\pi}_{t'}, \vec{\pi}_{t''}, \vec{Y}) \eqslantless^k \vec{\pi}_t \iff (\forall j \in [k])(\pi_{t',j}, \pi_{t'',j}, Y_j) \eqslantless \pi_{t,j}$$

In this way, our new concept makes it a relatively simple task to generalize a DP for connectivity-1 problems to a DP for connectivity-k problems (and facilitate the proof of correctness). There is a slight change in the way DPs are designed for each problem, but they follow the same principle.

Organization: We develop our techniques over several sections, and along the way, show non-trivial applications for various SNDP problems. Section 2 provides some notation and important definitions. Section 3 presents the new viewpoint for designing DP and presents a simple showcase by deriving (known) results. Section 4 presents algorithms for EC-SNDP. Lastly, Section 5 presents an approximation algorithm for group connectivity problems.

[6]This is due to the transitivity of edge-connectivity. Specifically, any vertices u, w that have k edge-disjoint paths connecting to the root r also have k edge-disjoint paths between themselves.

2 Preliminaries

Tree decomposition: Let G be any graph. A *tree decomposition* of G is a tree $\mathcal{T}$ with a collection of *bags* $\{X_t\}_{t \in V(\mathcal{T})} \subseteq 2^{V(G)}$ (i.e., each node of $\mathcal{T}$ is associated with a subset of nodes of $V(G)$) that satisfies the following properties:

- $V(G) = \bigcup_{t \in V(\mathcal{T})} X_t$
- For any edge $uv \in E(G)$, there is a bag X_t such that $u, v \in X_t$.
- For each vertex $v \in V(G)$, the collection of nodes t whose bags X_t contain v induces a connected subgraph of $\mathcal{T}$. That is, $\mathcal{T}[\{t \in V(\mathcal{T}) : v \in X_t\}]$ is a subtree of $\mathcal{T}$.

The treewidth of G, denoted $tw(G)$, is the minimum integer k for which there exists a tree decomposition $(\mathcal{T}, \{X_t\}_{t \in V(\mathcal{T})})$ such that $\max |X_t| \leq k + 1$ ($\max |X_t| - 1$ is the *width* of $\mathcal{T}$).

Fix a tree decomposition $(\mathcal{T}, \{X_t\})$ with the stated properties. For each node $t \in V(\mathcal{T})$, denote by $\mathcal{T}_t$ the subtree of $\mathcal{T}$ rooted at t. We also define G_t as the subgraph induced by $\mathcal{T}_t$; that is, $G_t = G\left[\bigcup_{t' \in \mathcal{T}_t} X_{t'}\right]$.

For each $v \in V$, let t_v denote the topmost node for which $v \in X_{t_v}$. For each $t \in V(\mathcal{T})$, we say that an edge $uv \in E(G)$ appears in the bag X_t if $u, v \in X_t$, and *only* if t is the topmost node in which this happens. We denote the edges inside the bag X_t by E_t. For a subset $\mathcal{S} \subseteq V(\mathcal{T})$, we define $X_{\mathcal{S}} := \bigcup_{t \in \mathcal{S}} X_t$.

We will use the following result, which shows that a tree decomposition of G of width $O(tw(G))$ is computable in time $O(2^{O(tw(G))}n)$.

Theorem 2.1 ([5])**.** *There is an algorithm that, given a graph G, runs in time $O(2^{O(tw(G))}n)$ and finds the tree decomposition $(\mathcal{T}, \{X_t\}_{t \in V(\mathcal{T})})$ such that $|X_t| \leq 5tw(G)$ for all t.*

In order to simplify notation, we assume that $\mathcal{T}$ is a binary tree. Furthermore, we require the height of $\mathcal{T}$ to be $O(\log n)$ in Section 5. The following lemma, based on the results of Bodlaender [4], summarizes the properties we assume.

Lemma 2.2 (in [12], based on [4])**.** *Given a tree decomposition $(\mathcal{T}', \{X'_t\}_{t \in V(\mathcal{T}')})$, we can transform into a tree decomposition $(\mathcal{T}, \{X_t\}_{t \in V(\mathcal{T})})$ with the following properties: (i) the height of $\mathcal{T}$ is at most $O(\log n)$; (ii) each bag X_t satisfies $|X_t| \leq O(w)$; (iii) every leaf bag has no edges ($E_t = \emptyset$ for leaf $t \in \mathcal{T}$); (iv) every non-leaf has exactly 2 children. Furthermore, this transformation is runs in linear time.*

←D

[7]

Connection sets and operators: Let $S \subseteq V$. A connection set Λ over S is a subset of $S \times S$ which will be used to list pairs that are connected via a path, i.e., $(u, v) \in \Lambda$ if there is a path connecting u to v. [8]

←D

[7] ⇒ Yes? –Daniel

[8] ⇒ I made the fix suggested by the reviewer, but I am not sure if it's the best way to do it. –Daniel

Let Λ be a connection set over S. The *transitive closure* operator, denoted by $\mathrm{tc}(\cdot)$, is defined naturally such that $\mathrm{tc}(\Lambda)$ contains all pairs (w, w') for which there is a sequence $(w = w_0, w_1, \ldots, w_q = w')$ and $(w_i, w_{i+1}) \in \Lambda$ for all $i < q$. Let $S' \subseteq S$. The *projection operator* "$\big|$" is defined such that $\Lambda\Big|_{S'} = \Lambda \cap (S' \times S')$. See Figure 1 for an illustration.

Given two connection sets Λ_1 of S_1 and Λ_2 of S_2, the union $\Lambda_1 \cup \Lambda_2$ is a connection set over $S_1 \cup S_2$.

3 New Key Concept: Global $\Leftrightarrow$ Local Checking for DP

This section introduces the key concept devised for handling all our problems systematically.

High-level intuition: Our DP will try to maintain a pair of local and global information about connectivity in the graph. Roughly speaking, a local connection set Γ_t for t (more precisely, for X_t) gives information about connectivity of the solution inside the subgraph G_t (the subgraph induced in subtree $\mathcal{T}_t$), while the other connection Δ_t for t gives information about connectivity of the global solution (i.e., the solution for the whole graph G).

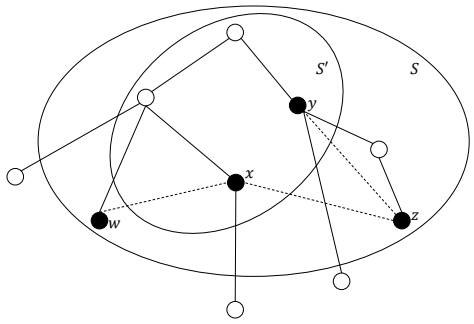

Figure 1: Connection sets, transitive closures and projections.
$\Lambda = \{(w, x), (x, z), (z, y)\}$ (dotted connections).
$\mathrm{tc}(\Lambda) = \{(w, x), (x, y), (y, z), (w, y), (x, z), (w, z)\}$.
$\mathrm{tc}(\Lambda)\Big|_{S'} = \{(x, y)\}$.

For instance, if we have a tentative solution $Y \subseteq E(G)$, we would like to have the information about the reachability of Y inside each bag, i.e., $\Delta_t = \mathrm{tc}(Y)\Big|_t$, so we could check the reachability between u and v simply by looking at whether $(u, v) \in \Delta_t$. However, a DP that is executing at node t may not have this global information, and this often leads to complicated rules to handle this situation.

We observe that global information can be passed along to all cells in the DP with simple local rules so that checking whether the connectivity requirements are satisfied can be done locally inside each DP cell. In the next section, we elaborate on this more formally.

Equivalence: One could imagine having a DP cell $c[t, \Gamma_t, \Delta_t]$ for all possible connection sets Γ_t, Δ_t, which makes a decision on Y_t, the set of edges bought by the solution when executing the DP. The roles of Γ_t and Δ_t are to give information about local and global reachability, respectively. We are seeking a solution Y that is "consistent" with these profiles, where Y can be partitioned based on the tree $\mathcal{T}$ as $Y_t = Y \cap E_t$.

Given Y, we say that the pairs $\{(\Gamma_t, \Delta_t)\}_{t \in V(\mathcal{T})}$ satisfy the *local* (resp., *global*) *connectivity definition* if, for every node $t \in V(\mathcal{T})$ (having left and right children as t' and t'', respectively),

Local

$$\Gamma_t := \begin{cases} \emptyset & \text{if } t \text{ is a leaf of } \mathcal{T} \\ \text{tc}\left(\Gamma_{t'} \cup \Gamma_{t''} \cup Y_t\right)\Big|_t & \text{otherwise} \end{cases}$$

$$\Delta_t := \begin{cases} \Gamma_t & \text{if } t = \text{root}(\mathcal{T}) \\ \text{tc}(\Delta_{p(t)} \cup \Gamma_t)\Big|_t & \text{otherwise} \end{cases}$$

Global

$$\Gamma_t := \text{tc}\left(Y \cap E(G_t)\right)\Big|_t$$

$$\Delta_t := \text{tc}\left(Y\right)\Big|_t$$

where the projection operator on X_t is simplified as $|_t$.

The main idea is that the local connectivity definition gives us "local" rules that enforce consistency of consecutive bags, and this would be suitable for being embedded into a DP. The global connectivity rules, however, are not easily encoded into DP, but it is easy to argue intuitively and formally about their properties. The following lemma (proof in Appendix A) shows that the local and global connectivity definitions are, in fact, equivalent.

Lemma 3.1. *Let $Y_t \subseteq E_t$ be a subset of edges and (Γ_t, Δ_t) a pair of connectivity sets for every $t \in V(\mathcal{T})$. Then, the pairs (Γ_t, Δ_t) satisfy the local connectivity definition iff they satisfy the global connectivity definition.*

The notions of local and global connectivity, as well as the equivalence between them can be generalized both for the edge-connectivity version with vertex-costs as well as vertex-connectivity with vertex-costs. We defer the details of this generalization to Appendix E.

A warmup application: steiner trees. We now show an approach that allows us to solve the Steiner Tree problem exactly in $nw^{O(w)}$ time, given a tree decomposition of width w. We remark that the best known algorithm due to Cygan et al. [18] runs in time $2^{O(w)}n$. In this problem, we are given graph $G = (V, E)$ with edge-costs and terminals $T = \{v_1, \ldots, v_h\}$, and the goal is to find a min-cost subset $E^* \subseteq E$ that connects all the terminals. For simplicity, we denote v_1 by "root" r, and add r to every bag. The goal is now to connect the root to all other terminals in $T \setminus r$.

Our DP table has a cell $c[t, \Gamma, \Delta]$ for every node $t \in V(\mathcal{T})$ and every pair of connection sets (Γ, Δ) for t. We initialize the DP table by setting $c[t, \Gamma, \Delta]$ for all the leaf nodes, and setting certain cells as invalid (by setting $c[t, \Gamma, \Delta] = \infty$):

- For every leaf node t and every pair (Γ, Δ), we set $c[t, \Gamma, \Delta] = 0$ if $\Gamma = \emptyset$ and $c[t, \Gamma, \Delta] = \infty$ otherwise.
- We mark the cells $(\text{root}(\mathcal{T}), \Gamma, \Delta)$ as invalid if $\Gamma \neq \Delta$.
- Let $v_i \in T$ be one of the terminals, and $t \in V(\mathcal{T})$ a node. We mark a cell (t, Γ, Δ) as invalid if $v_i \in X_t$ but $(r, v_i) \notin \Delta$.

For all other cells, we compute the result from their children. Let t be a node with left-child t' and right-child t''. Let $(\Gamma, \Delta), (\Gamma', \Delta'), (\Gamma'', \Delta'')$ be pairs of connection sets for t, t', t'', respectively, and $Y_t \subseteq E_t$. We say that (Γ, Δ) is consistent with $((\Gamma', \Delta'), (\Gamma'', \Delta''))$ via Y_t (abbreviated by the notation $(\Gamma, \Delta) \xleftrightarrow{Y_t} ((\Gamma', \Delta'), (\Gamma'', \Delta''))$ if

$$\Gamma = \text{tc}(\Gamma' \cup \Gamma'' \cup Y_t)\Big|_t \qquad \Delta' = \text{tc}(\Delta \cup \Gamma')\Big|_{t'} \qquad \Delta'' = \text{tc}(\Delta \cup \Gamma'')\Big|_{t''}$$

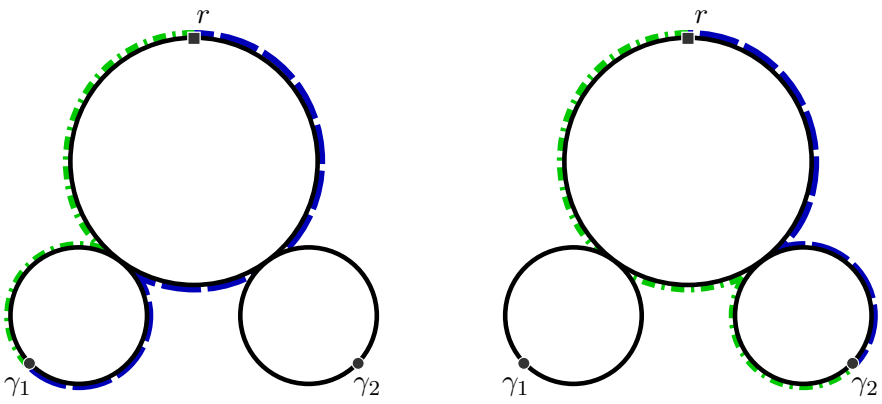

Figure 2: Example of different partitions of edges into paths from demands γ_1, γ_2. On the left (resp., right), two paths from r to γ_1 (resp., γ_2)

Now, for any choice of valid DP cells (t, Γ_t, Δ_t) and edge subsets $Y_t \subseteq E_t$ for every $t \in V(\mathcal{T})$ (notice that a DP solution uses precisely one cell per node t), we apply Lemma 3.1 to conclude that since the local connectivity definition is satisfied for (Γ_t, Δ_t) pairs, so does the global connectivity definition. Since, for every valid DP cell (t, Γ, Δ) such that t contains a terminal v_i, $(r, v_i) \in \Delta$, we conclude that every terminal is connected to the root in the solution $Y = \bigcup_{t \in V(\mathcal{T})} Y_t$. Thus, the cost of the optimum solution can be found at one of the cells $c[\text{root}(\mathcal{T}), \Gamma, \Delta]$ (with $\Gamma = \Delta$).

Conversely, given a solution $F \subseteq E(G)$, we can define $F_t := F \cap E_t$, and a pair (Γ_t, Δ_t) for every $t \in V(\mathcal{T})$, using the global connectivity definition. Lemma 3.1 implies that the pairs (Γ_t, Δ_t) satisfy the local connectivity definition, and therefore define valid DP cells (notice that for every terminal v_i , F connects v_i to the root, so $(r, v_i) \in \Delta_t$ for every $t \in V(\mathcal{T})$ such that $v_i \in X_t$). We thus establish that for every valid DP solution there is a corresponding feasible solution $F \subseteq E(G)$, and vice-versa.

4 Extension to High Connectivity

This section shows how to apply our framework to problems with high connectivity requirements. We focus on edge-connectivity and leave the case of vertex-connectivity to Appendix E.

When solving a problem in a high connectivity setting, there may be a requirement of k disjoint paths. In particular, for each demand pair (u, v), there must be $k(u, v)$ disjoint paths in the solution. So we could start naturally with a profile of the form:

$$(\Gamma_{t,1}, \ldots, \Gamma_{t,k})(\Delta_{t,1}, \ldots, \Delta_{t,k})$$

for each node $t \in V(\mathcal{T})$, and enforce the local consistency conditions for each coordinate. Here, each pair $(\Gamma_{t,j}, \Delta_{t,j})$ would correspond to connectivity in some subgraph $H_j \subseteq G$, where H_j serve the j-th path of the demand (u, v).

However, this idea does not work as a different demand pair, say (a, b), might use a path that belongs to different subgraphs H_j as defined above. In other words, the disjoint paths for the demand pair (a, b) might require a different partitioning of the solution set H. Figure 2 illustrates

a case in which different demand pairs use different partitions. Therefore, we need to enumerate all possible ways for the demands to "locally" partition the graph and use them to support all k disjoint paths for each one of them. This requires a more careful local consistency check between the DP cells.

We consider the Rooted EC-SNDP problem with edge-costs as an example to explain how to apply our framework in high-connectivity, that is, we will prove Theorem 1.3. For convenience, we add r to every bag. To avoid confusion, the root of the tree $\mathcal{T}$ will be referred explicitly as root($\mathcal{T}$).

The organization of this section is as follows. In Section 4.1, we explain the setup of the cells of the DP table and a high-level intuition about how the DP works. In Section 4.2 we describe the algorithm, in particular, how to compute the values of the DP table. We leave the discussion of its correctness and running time to Appendix B.2.

4.1 Profiles

As in any standard dynamic programming approach based on tree decomposition, we have a profile for each node t, which tries to solve the subproblem restricted to G_t in some way.

Let $t \in V(\mathcal{T})$. A *connection profile* for t is a k-tuple $\vec{\Gamma} = (\Gamma_1, \Gamma_2, \ldots, \Gamma_k)$ such that $\Gamma_i \subseteq X_t \times X_t$. Let $\mathcal{X}_t$ be the set of all connection profiles for t. A profile Ψ of node t is a collection of pairs of connection profiles $(\vec{\Gamma}, \vec{\Delta})$, i.e., $\Psi \subseteq \mathcal{X}_t \times \mathcal{X}_t$. A partial solution $F \subseteq E(G_t)$ is said to be *consistent* with profile Ψ for t if, for all $(\vec{\Gamma}, \vec{\Delta}) \in \Psi$,

- For each $(u, v) \in \Gamma_i$ and $(a, b) \in \Gamma_j$ for $i \neq j$, there are paths $P_{uv}, P_{ab} \subseteq F$ connecting the respective vertices in G_t such that P_{uv} and P_{ab} are edge-disjoint.
- There is a global solution $F' \supseteq F$ such that, for each $(u, v) \in \Delta_i$ and $(a, b) \in \Delta_j$ for $i \neq j$, there are paths $Q_{uv}, Q_{ab} \subseteq F'$ connecting the respective vertices in G such that P_{uv} and P_{ab} are edge-disjoint.

In other words, a solution consistent with a profile must "implement" all connectivity requirements by $\vec{\Gamma}$ and must be extensible to satisfy $\vec{\Delta}$.

Passing down both local and global requirements in the DP table leads to a clean and simple DP algorithm. Our DP table has a cell $c[t, \Psi]$ for each $t \in V(\mathcal{T})$ and each profile Ψ for t. This cell tentatively stores the optimal cost of a solution consistent with profile Ψ.

4.2 The DP

Valid cells: Some table entries do not correspond to valid solutions of the problem, so we mark them as invalid and remove them from consideration (another way to think about this is that we initialize $c[t, \Psi] = \infty$ for all invalid cells), i.e., the following cells are invalid:

- Any leaf that has non-empty connectivity requirements is invalid. That is, $c[t, \Psi] = 0$ if $\Psi \subseteq \{(\emptyset, \ldots, \emptyset)\} \times \mathcal{X}_t$; otherwise, $c[t, \Psi] = \infty$.

- Any cell that cannot be extended into a feasible solution is invalid. If there is no pair $(\vec{\Gamma}, \vec{\Delta}) \in \Psi$ such that $(r, \gamma_i) \in \Delta_j$ for all $j \in [k_i]$, then there are fewer than k_i edge-disjoint paths between r and γ_i, and therefore the cell is invalid, so $c[t_{\gamma_i}, \Psi] = \infty$.
- The node root$(\mathcal{T})$ together with profile Ψ is an invalid cell if there is a pair $(\vec{\Gamma}, \vec{\Delta}) \in \Psi$ such that $\Gamma_j \neq \Delta_j$ for some j. In this case, we set $c[\text{root}(\mathcal{T}), \Psi] = \infty$.

Lemma 4.1. *For every $t \in \mathcal{T}$, there are at most $\exp(w^{O(wk)})$ many valid cells (t, Ψ).*

We remark that, though the solution can be partitioned differently for each demand, the number of different partitions of a solution still depends on its size. Therefore, once we look at the partitions for each bag X_t, the number of possibilities is bounded by a function of w and k. [9] ←D

DP computation: For all other cells, we compute their values from the values of their children. Let t be a node with left-child t' and right-child t''. Let Ψ, Ψ', Ψ'' be their profiles respectively, and $Y_t \subseteq E_t$. We say that Ψ is consistent with (Ψ', Ψ'') via Y_t (abbreviated by $\Psi \xleftrightarrow{Y_t} (\Psi', \Psi'')$) if the following conditions are satisfied. For each pair $(\vec{\Gamma}, \vec{\Delta}) \in \Psi$, there are $(\vec{\Gamma}', \vec{\Delta}') \in \Psi'$ and $(\vec{\Gamma}'', \vec{\Delta}'') \in \Psi''$, together with a partition of Y_t into $Y_1 \cup Y_2 \cup \ldots \cup Y_k$ such that, for every $j \in [k]$,

$$\Gamma_j = \text{tc}(\Gamma'_j \cup \Gamma''_j \cup Y_j)\Big|_t \qquad \Delta'_j = \text{tc}(\Delta_j \cup \Gamma'_j)\Big|_{t'} \qquad \Delta''_j = \text{tc}(\Delta_j \cup \Gamma''_j)\Big|_{t''}$$

Similarly, for any $(\vec{\Gamma}', \vec{\Delta}') \in \Psi'$ (resp., $(\vec{\Gamma}'', \vec{\Delta}'') \in \Psi''$) there are $(\vec{\Gamma}'', \vec{\Delta}'') \in \Psi''$ (resp., $(\vec{\Gamma}', \vec{\Delta}') \in \Psi'$) plus $(\vec{\Gamma}, \vec{\Delta}) \in \Psi$ and some partition of Y_t, satisfying similar conditions as above.

Then the value of $c[t, \Psi]$ can be defined recursively among valid cells:

$$c[t, \Psi] = \min_{\Psi \xleftrightarrow{Y_t} (\Psi', \Psi'')} (c[t', \Psi'] + c[t'', \Psi''] + c(Y_t))$$

The final solution can be computed as $\min\left\{c[\text{root}(\mathcal{T}), \Psi] \mid (\forall (\vec{\Gamma}, \vec{\Delta}) \in \Psi)(\forall j \in [k])\Gamma_j = \Delta_j\right\}$. The correctness of this DP is deferred to Appendix B.1.

5 Algorithms for Restricted Group SNDP

In Section 4, we showed how to find the optimum solution to the EC-SNDP by solving a DP. This is only possible because we know the exact demands that correspond to a subproblem, and we mark cells as invalid if these demands are not met. In the Restricted Group SNDP, each group can be connected in subproblems corresponding to different part of the trees, and hence it is no longer possible to solve the DP in a standard way.

We do, however, have sufficient technical tools to prove Theorem 1.1. Instead of simply solving the DP, we turn the DP table into a tree instance of a variant of GST and, in a second step, apply randomized rounding to obtain a polylogarithmic approximation to the problem. This corresponds to the process of finding a solution in the DP, with the additional constraint that all the demands are met. ←D

[9] ⇒ I don't like this that much, but I don't see how else to do it –Daniel

Tree Instance: We start by showing how to transform the DP table into a tree $\widetilde{\mathcal{T}}$, where we can solve a variant of the group Steiner tree problem. The following theorem formalizes this transformation, and we dedicate the rest of this section to proving the theorem. For convenience, we add a dummy node t_S with $X_{t_S} = \emptyset$ as the parent of the root node.

Theorem 5.1. *Given a graph G rooted at r with treewidth w and groups $S_i \subseteq V$, there is a tree $\widetilde{\mathcal{T}}$ with groups $\widetilde{S}_i$ and a set of accepted solutions $\widetilde{X}$ such that: (i) the size of $\widetilde{\mathcal{T}}$ is $n^{w^{O(wk)}}$; (ii) for every $F \subseteq E(G)$, there is $X \in \widetilde{X}$ (and vice-versa) such that $c(F) = c(X)$ and, for every $i \in [h]$, F k_i-connects r to $v \in S_i$ iff X connects* $\mathrm{root}(\widetilde{T})$ *to $\tilde{t} \in \widetilde{S}_i$.*

For each cell of the DP table introduced in Section 4.2, we create a node in $\widetilde{\mathcal{T}}$. Namely, we create a vertex $\tilde{t}[t, \Psi]$ for every node $t \in V(\mathcal{T})$ and $\Psi \subseteq \mathcal{X}_t \times \mathcal{X}_t$. The root of the tree is $\tilde{t}[t_S, \{(\emptyset^k, \emptyset^k)\}]$ (this is the only connection profile for t_S). For a node $t \in \mathcal{T}$ with children t', t'', we add connecting nodes $\tilde{t}_c[t, \Psi, \Psi', \Psi'', Y_t]$ connected to the nodes $\tilde{t}[t, \Psi]$, $\tilde{t}[t', \Psi']$, $\tilde{t}[t'', \Psi'']$, for every $Y_t \subseteq E_t$, if $\Psi \xleftrightarrow{Y_t} (\Psi', \Psi'')$. If there is only one child, the connecting node has degree 2, and we consider that $\Psi'' = \{(\emptyset^k, \emptyset^k)\}$ for the purpose of describing the algorithm. An edge from $\tilde{t}[t, \Psi]$ to $\tilde{t}_c[t, \Psi, \Psi', \Psi'', Y_t]$ is labeled with the set of edges Y_t and is assigned cost $c(Y_t)$. All other edges in the instance have cost 0.

Notice that, at this point, $\widetilde{\mathcal{T}}$ is not a tree, but we can turn it into one by making copies of the nodes as required. Specifically, we process the tree in a bottom-up fashion: for each node $\tilde{t}$, we make the same number of copies of $\tilde{t}$ and its descendants as there are incoming edges of $\tilde{t}$ such that each edge is incident to a different copy. In this manner, all the copies of $\tilde{t}$ are now the roots of subtrees, which are disjoint.

For convenience, we denote by $\tilde{t}[t, \Psi]$ (resp., $\tilde{t}_c[t, \Psi, \Psi', \Psi'', Y_t]$) any copy of the original node; when we need to distinguish copies, we denote by $\mathrm{copies}(\tilde{t})$ the set of all copies of a node $\tilde{t}$.

The final step in our construction is to prune the tree by removing nodes that cannot be reached or that represent choices that cannot be part of a feasible solution. To do that, it is sufficient to apply the following rules to exhaustion: (i) remove $\tilde{t} \in \widetilde{\mathcal{T}}$ if it is not connected to the root; (ii) remove a connecting node if one of its children was removed; (iii) remove $\tilde{t}[t, \Psi]$ if it is a leaf node but $\Psi \not\subseteq \{(\emptyset, \ldots, \emptyset)\} \times \mathcal{X}_t$ (i.e. $\vec{\Gamma} \neq (\emptyset, \ldots, \emptyset)$ for some $(\vec{\Gamma}, \vec{\Delta}) \in \Psi$).

We can now restate the goal of the problem in terms of $\widetilde{\mathcal{T}}$: we want to find nodes $\tilde{t}[t, \Psi_t]$ and edge sets $Y_t \subseteq E_t$ for every node $t \in V(\mathcal{T})$, such that $(\Psi_{t'}, \Psi_{t''}) \xleftrightarrow{Y_t} \Psi_t$ for all non-leaf node t with children t', t''. Further, for every group S_i, there must be a vertex $\gamma_i \in S_i$ and a partition of $Y := \bigcup_{t \in V(\mathcal{T})} Y_t$ into k_i sets, such that each contains a path from r to γ_i.

The set of nodes $\{\tilde{t}[t, \Psi_t]\}_{t \in V(\mathcal{T})}$ with the respective connecting nodes $\tilde{t}[t, \Psi_t, \Psi_{t'}, \Psi_{t''}, Y_t]$ (for all non-leaf nodes $t \in V(\mathcal{T})$) induces a tree $\widetilde{T}$ in $\widetilde{\mathcal{T}}$. We say that such a tree $\widetilde{T}$ is *valid* if every node $\tilde{t}[t, \Psi] \in V(\widetilde{T})$ has exactly one child in the graph (or none if t is a leaf), and every connecting node $\tilde{t}_c[t, \Psi, \Psi', \Psi'', Y_t]$ in $\widetilde{T}$ has full-degree, i.e., all its neighbors are in the solution as well.

For every group S_i, we define $\widetilde{S}_i$ as follows: for every $v \in S_i$ and every $\Psi \in \mathcal{X}_t \times \mathcal{X}_t$, every element

[10] ⇒ I rewrote this a bit and added more "overview", as reviewer 2 wanted. THe old version is commented in case you want it –Daniel

of copies($\tilde{t}[t_v, \Psi]$) is in $\widetilde{S}_i$ if there is $(\vec{\Gamma}, \vec{\Delta}) \in \Psi$ such that $(r, v) \in \Delta_j$ for all $j \in [k_i]$.

The size of the instance follows by considering its height and maximum degree: the maximum degree of $\widetilde{\mathcal{T}}$ is $\exp(w^{O(wk)})$ by Lemma 4.1, and there is a tree decomposition of height $O(\log n)$ by Lemma 2.2, which implies that $\text{height}(\widetilde{\mathcal{T}}) = O(\log n)$. We conclude that $|\widetilde{\mathcal{T}}| = n^{w^{O(wk)}}$.

The correctness of the reduction follows from the correctness of the DP for Rooted EC-SNDP, and its proof is left to Appendix B.1.

Algorithm: We now show how to obtain a valid tree $\widetilde{T}$, given $\widetilde{\mathcal{T}}$. Let T^* be the min-cost valid tree in $\widetilde{\mathcal{T}}$ that connects all the groups $\{\widetilde{S}_i\}_{i \in [h]}$. Chalermsook et al. [12] showed that it is possible to find a valid tree $\widetilde{T}$ with expected cost $c(T^*)$, but whose probability of covering a group is just $O(1/\text{height}(\widetilde{\mathcal{T}}))$.

Using this result, we can obtain valid trees $\widetilde{T}_1, \ldots, \widetilde{T}_\ell$, where $\ell = O(\log n \log h)$. We can then obtain solutions F_j that k-connect the same groups and have the same cost as $\widetilde{T}_j$, for all $j \in [\ell]$, and finally output the solution $F := \bigcup_{j \in [\ell]} F_j$. Since the expected cost of each $\tilde{T}_i$ is $c(T^*)$, the expected cost of F is $O(\log n \log h) c(T^*)$.

By sampling $c \log n \log h$ independent valid trees, for large enough c, we ensure that each group is covered with probability $(1/\log n)^\ell = 1/h^c$, and thus all the groups are covered with high probability (by union bound). For any group that is not covered, we can add minimum-cost edge-disjoint paths from the root (by reduction to min-cost flow) and hence ensure that every group is covered, without increasing the expected cost of the solution. [11] We conclude that the algorithm outputs a randomized $O(\log n \log h)$-approximation to the problem, with high probability. ←D

Acknowledgements. Part of this work was done while Parinya Chalermsook, Bundit Laekhanukit and Daniel Vaz were visiting the Simons Institute for the Theory of Computing. It was partially supported by the DIMACS/Simons Collaboration on Bridging Continuous and Discrete Optimization through NSF grant #CCF-1740425. Parinya Chalermsook is currently supported by European Research Council (ERC) under the European UnionâĂŹs Horizon 2020 research and innovation programme (grant agreement No 759557) and by Academy of Finland Research Fellows, under grant number 310415

References

[1] MohammadHossein Bateni, Mohammad Taghi Hajiaghayi, and Dániel Marx. Approximation schemes for steiner forest on planar graphs and graphs of bounded treewidth. *J. ACM*, 58(5):21:1–21:37, 2011.

[2] Mihir Bellare, Shafi Goldwasser, Carsten Lund, and A. Russeli. Efficient probabilistically checkable proofs and applications to approximations. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing, May 16-18, 1993, San Diego, CA, USA*, pages 294–304, 1993.

[11] ⇒ Added this to clarify that a feasible solution is always returned. –Daniel

[3] André Berger and Michelangelo Grigni. Minimum weight 2-edge-connected spanning subgraphs in planar graphs. In *Automata, Languages and Programming, 34th International Colloquium, ICALP 2007, Wroclaw, Poland, July 9-13, 2007, Proceedings*, pages 90–101, 2007.

[4] Hans L. Bodlaender. NC-algorithms for graphs with small treewidth. In Jan van Leeuwen, editor, *Graph-Theoretic Concepts in Computer Science, 14th International Workshop, WG '88, Amsterdam, The Netherlands, June 15-17, 1988, Proceedings*, volume 344 of *Lecture Notes in Computer Science*, pages 1–10. Springer, 1988.

[5] Hans L. Bodlaender, Pål Grønås Drange, Markus S. Dregi, Fedor V. Fomin, Daniel Lokshtanov, and Michal Pilipczuk. A c^k n 5-approximation algorithm for treewidth. *SIAM J. Comput.*, 45(2):317–378, 2016.

[6] Richard B. Borie, R. Gary Parker, and Craig A. Tovey. Automatic generation of linear-time algorithms from predicate calculus descriptions of problems on recursively constructed graph families. *Algorithmica*, 7(5&6):555–581, 1992.

[7] Glencora Borradaile, Erik D. Demaine, and Siamak Tazari. Polynomial-time approximation schemes for subset-connectivity problems in bounded-genus graphs. *Algorithmica*, 68(2):287–311, 2014.

[8] Glencora Borradaile, Philip N. Klein, and Claire Mathieu. An $O(n \log n)$ approximation scheme for steiner tree in planar graphs. *ACM Trans. Algorithms*, 5(3):31:1–31:31, 2009.

[9] Glencora Borradaile, Philip N. Klein, and Claire Mathieu. A polynomial-time approximation scheme for euclidean steiner forest. *ACM Trans. Algorithms*, 11(3):19:1–19:20, 2015.

[10] Glencora Borradaile and Baigong Zheng. A PTAS for three-edge-connected survivable network design in planar graphs. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2017, August 16-18, 2017, Berkeley, CA, USA*, pages 3:1–3:13, 2017.

[11] İSMET Esra Büyüktahtakin. *Dynamic Programming Via Linear Programming.* John Wiley & Sons, Inc., 2010.

[12] Parinya Chalermsook, Syamantak Das, Bundit Laekhanukit, and Daniel Vaz. Beyond metric embedding: Approximating group steiner trees on bounded treewidth graphs. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 737–751, 2017.

[13] Parinya Chalermsook, Fabrizio Grandoni, and Bundit Laekhanukit. On survivable set connectivity. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 25–36, 2015.

[14] Joseph Cheriyan and Adrian Vetta. Approximation algorithms for network design with metric costs. *SIAM J. Discrete Math.*, 21(3):612–636, 2007.

[15] Julia Chuzhoy and Sanjeev Khanna. Algorithms for single-source vertex connectivity. In *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008, October 25-28, 2008, Philadelphia, PA, USA*, pages 105–114, 2008.

[16] Julia Chuzhoy and Sanjeev Khanna. An o(k^3log n)-approximation algorithm for vertex-connectivity survivable network design. *Theory of Computing*, 8(1):401–413, 2012.

[17] Bruno Courcelle. The monadic second-order logic of graphs. i. recognizable sets of finite graphs. *Inf. Comput.*, 85(1):12–75, 1990.

[18] Marek Cygan, Jesper Nederlof, Marcin Pilipczuk, Michal Pilipczuk, Johan M. M. van Rooij, and Jakub Onufry Wojtaszczyk. Solving connectivity problems parameterized by treewidth in single exponential time. In *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 150–159, 2011.

[19] Artur Czumaj, Michelangelo Grigni, Papa Sissokho, and Hairong Zhao. Approximation schemes for minimum 2-edge-connected and biconnected subgraphs in planar graphs. In *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2004, New Orleans, Louisiana, USA, January 11-14, 2004*, pages 496–505, 2004.

[20] Daniela Pucci de Farias and Benjamin Van Roy. Approximate dynamic programming via linear programming. In *Advances in Neural Information Processing Systems 14 [Neural Information Processing Systems: Natural and Synthetic, NIPS 2001, December 3-8, 2001, Vancouver, British Columbia, Canada]*, pages 689–695, 2001.

[21] F d'Epenoux. A probabilistic production and inventory problem. *Management Science*, 10(1):98–108, 1963.

[22] Naveen Garg, Goran Konjevod, and R. Ravi. A polylogarithmic approximation algorithm for the group steiner tree problem. *J. Algorithms*, 37(1):66–84, 2000.

[23] Anupam Gupta, Ravishankar Krishnaswamy, and R. Ravi. Tree embeddings for two-edge-connected network design. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*, pages 1521–1538, 2010.

[24] Anupam Gupta, Kunal Talwar, and David Witmer. Sparsest cut on bounded treewidth graphs: algorithms and hardness results. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 281–290, 2013.

[25] Eran Halperin and Robert Krauthgamer. Polylogarithmic inapproximability. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing, June 9-11, 2003, San Diego, CA, USA*, pages 585–594, 2003.

[26] Kamal Jain. A factor 2 approximation algorithm for the generalized steiner network problem. *Combinatorica*, 21(1):39–60, 2001.

[27] Rohit Khandekar, Guy Kortsarz, and Zeev Nutov. Approximating fault-tolerant group-steiner problems. *Theor. Comput. Sci.*, 416:55–64, 2012.

[28] Bundit Laekhanukit. Parameters of two-prover-one-round game and the hardness of connectivity problems. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 1626–1643, 2014.

[29] Bundit Laekhanukit. An improved approximation algorithm for the minimum cost subset k-connected subgraph problem. *Algorithmica*, 72(3):714–733, 2015.

[30] Alan S Manne. Linear programming and sequential decisions. *Management Science*, 6(3):259–267, 1960.

[31] R. Kipp Martin, Ronald L. Rardin, and Brian A. Campbell. Polyhedral characterization of discrete dynamic programming. *Operations Research*, 38(1):127–138, 1990.

[32] Zeev Nutov. Approximating minimum-cost connectivity problems via uncrossable bifamilies. *ACM Trans. Algorithms*, 9(1):1:1–1:16, 2012.

[33] Marcin Pilipczuk, Michal Pilipczuk, Piotr Sankowski, and Erik Jan van Leeuwen. Subexponential-time parameterized algorithm for steiner tree on planar graphs. In *30th International Symposium on Theoretical Aspects of Computer Science, STACS 2013, February 27 - March 2, 2013, Kiel, Germany*, pages 353–364, 2013.

[34] David P. Williamson, Michel X. Goemans, Milena Mihail, and Vijay V. Vazirani. A primal-dual approximation algorithm for generalized steiner network problems. *Combinatorica*, 15(3):435–454, 1995.

A Details of the Global $\Leftrightarrow$ Local Checking for DP

In this section, we will prove a generalized version of Lemma 3.1, that works for edge-connectivity and vertex-connectivity, both with edge and vertex costs. In vertex-connectivity problems, we are interested in finding internally disjoint paths. In order to handle this setting, we introduce a modified version of transitive closure.

For a set of vertices Z and a set of edges S, we denote by $\mathrm{tc}^*_Z(S)$ the set of all pairs (u, v) such that there is a u-v-path in the graph $(Z \cup \{u, v\}, S)$, that is, a path whose internal vertices are in Z, and whose edges are in S. Formally,

$$\mathrm{tc}^*_Z(S) = \Big\{(u, v) \;\Big|\; \exists w_1, \ldots, w_\ell \in Z, \forall i \in [\ell - 1], (u, w_1), (w_\ell, v), (w_i, w_{i+1}) \in S\Big\}$$

We then keep track, for every node, of which vertices in the bag are allowed to be used in the solution, that is, we use triples (Z, Γ^*, Δ^*), instead of the previously used pairs (Γ, Δ).

Let $W_t \subseteq X_t \setminus X_{p(t)}$ for every $t \in V(\mathcal{T})$, $W = \bigcup_{t \in V(\mathcal{T})} W_t$, $Y_t \subseteq E_t$ for every $t \in V(\mathcal{T})$, $Y = \bigcup_{t \in V(\mathcal{T})} Y_t$, and a triple $(Z_t, \Gamma^*_t, \Delta^*_t)$ for every $t \in V(\mathcal{T})$. For the purposes of this section, we introduce the following definitions for local and global connectivity.

We say that the triples $(Z_t, \Gamma^*_t, \Delta^*_t)$ satisfy the *local (resp. global) connectivity definition* if, for every node $t \in V(\mathcal{T})$,

Local

$$Z_t := \begin{cases} W_t & \text{if } t = \text{root}(\mathcal{T}) \\ \left(Z_{p(t)} \cup W_t\right) \cap V_t & \text{otherwise} \end{cases}$$

$$\Gamma^*_t := \begin{cases} \emptyset & \text{if } t \text{ is a leaf node} \\ \text{tc}^*_{Z_t}\left(\Gamma^*_{t'} \cup \Gamma^*_{t''} \cup Y_t\right)\Big|_t & \text{otherwise} \end{cases}$$

$$\Delta^*_t := \begin{cases} \Gamma^*_t & \text{if } t = \text{root}(\mathcal{T}) \\ \text{tc}^*_{Z_t}\left(\Delta^*_{p(t)} \cup \Gamma^*_t\right)\Big|_t & \text{otherwise} \end{cases}$$

Global

$$Z_t := W \cap X_t$$

$$\Gamma^*_t := \text{tc}^*_W\left(Y \cap E(G_t)\right)\Big|_t$$

$$\Delta^*_t := \text{tc}^*_W\left(Y\right)\Big|_t$$

We then prove the following lemma, proving that the given local and global connectivity definitions are equivalent.

Lemma A.1. *Let $W_t \subseteq X_t \setminus X_{p(t)}$ be a subset of vertices, $Y_t \subseteq E_t$ a subset of edges and $(Z, \Gamma^*_t, \Delta^*_t)$ a triple of profiles for every $t \in V(\mathcal{T})$.*

Then, the triples $(Z_t, \Gamma^*_t, \Delta^*_t)$ satisfy the local connectivity definition iff they satisfy the global connectivity definition.

Before proving the lemma, we show how Lemma 3.1 follows. We will prove that, if we fix $W_t = X_t \setminus X_{p(t)}$ and $Z_t = X_t$, the definitions of Lemmas 3.1 and A.1 are equivalent.

For this, it is sufficient to see that $\text{tc}^*_W(S) = \text{tc}^*_{V(G)}(S) = \text{tc}(S)$, and that the common vertices in $\Gamma^*_{t'}$, $\Gamma^*_{t''}$, and Y_t are all in X_t, thus

$$\text{tc}^*_{Z_t}\left(\Gamma^*_{t'} \cup \Gamma^*_{t''} \cup Y_t\right)\Big|_t = \text{tc}\left(\Gamma^*_{t'} \cup \Gamma^*_{t''} \cup Y_t\right)\Big|_t$$

Similarly, since $\Delta^*_{p(t)}$ and Γ^*_t only intersect inside $X_t \times X_t$,

$$\text{tc}^*_{Z_t}\left(\Delta^*_{p(t)} \cup \Gamma^*_t\right)\Big|_t = \text{tc}\left(\Delta^*_{p(t)} \cup \Gamma^*_t\right)\Big|_t$$

We conclude that when $Z_t = X_t$, $\Gamma^*_t = \Gamma_t$ and $\Delta^*_t = \Delta_t$, the proof follows.

The following technical lemma will be useful when proving Lemma A.1.

Lemma A.2 (Path Lemma)**.** *Let G be any graph and $\mathcal{T}$ be a tree decomposition of G. Let $t \in V(\mathcal{T})$ and P be a path of length at least 2 whose endpoints x, y are the only vertices of P in t, that is, $V(P) \cap X_t \subseteq \{x, y\}$.*

Then there is a connected (subtree) component $\mathcal{T}'$ in $\mathcal{T} \setminus t$ such that, for any edge $ab \in E(P)$, $\mathcal{T}'$ has a node t' that contains ab, i.e., every edge $ab \in E_{t'}$ for some node $t' \in V(\mathcal{T}')$.

Proof. We provide a simple proof by contradiction. Assume that there are two consecutive edges, $ab, bc \in E(P)$ that are in different connected components of $\mathcal{T} \setminus t$ (otherwise, all edges must be in the same component). Since the set of nodes whose bags contain b must be connected in $\mathcal{T}$ but is not connected in $\mathcal{T} \setminus \{t\}$, $b \in X_t$, and we reach a contradiction. □

Proof of Lemma A.1. We remark that the function tc^* shares some properties with the usual definition of transitive closure, which are used throughout the proof:

Observation A.3. *The function* tc^* *satisfies the following properties:*

- $\mathrm{tc}^*_Z(\mathrm{tc}^*_Z(Y)) = \mathrm{tc}^*_Z(Y)$
- $\mathrm{tc}^*_{Z'}(Y') \subseteq \mathrm{tc}^*_Z(Y)$ *if* $Z' \subseteq Z$, $Y' \subseteq Y$

Equivalence for Z_t: We prove that the two definitions for Z are equivalent by induction on the depth of the node. At the root, we have that $W_{\mathrm{root}(\mathcal{T})} = W \cap X_{\mathrm{root}(\mathcal{T})}$, so the equivalence holds.

For the induction step, let $t \in V(\mathcal{T})$ be a node other than the root. Then

$$\begin{aligned}\Big(Z_{p(t)} \cup W_t\Big) \cap X_t &= \Big(Z_{p(t)} \cap X_t\Big) \cup (W_t \cap X_t) \\ &\subseteq (W \cap X_t) \cup (W \cap X_t) \\ &= W \cap X_t\end{aligned}$$

The second step follows from the induction hypothesis, as well as the definition of W. We now prove the converse inclusion.

$$\begin{aligned}W \cap X_t &= \Big(W \cap X_{p(t)} \cap X_t\Big) \cup \Big(W \cap \Big(X_t \setminus X_{p(t)}\Big)\Big) \\ &\subseteq \Big(Z_{p(t)} \cap X_t\Big) \cup W_t \\ &= \Big(Z_{p(t)} \cup W_t\Big) \cap X_t\end{aligned}$$

We use the induction hypothesis, as well as the fact that $W_t \subseteq X_t$.

Equivalence for Γ^*_t: We prove the statement by induction on the height of a node t. Since $E(G_t) = \emptyset$, both definitions are equivalent for every leaf t.

Let t be any node. By the induction hypothesis, $\Gamma^*_{t'}, \Gamma^*_{t''} \subseteq \mathrm{tc}^*_W\left(Y \cap E(G_t)\right)$. Therefore,

$$\mathrm{tc}^*_{Z_t}\left(\Gamma^*_{t'} \cup \Gamma^*_{t''} \cup Y_t\right)\Big|_t \subseteq \mathrm{tc}^*_W\left(Y \cap E(G_t)\right)\Big|_t$$

Here, we use that $Z_t = W \cap X_t \subseteq W$.

To prove the converse inclusion, let $(u, v) \in \mathrm{tc}^*_W\left(Y \cap E(G_t)\right)\Big|_t$. By definition, there must be a path p between u and v using internal vertices in W. Let $u = w_0, w_1, \ldots, w_\ell = v$ be all the vertices of p (in the correct order) that are also in X_t.

Each pair (w_i, w_{i+1}) is connected by a subpath of p. By Lemma A.2, (w_i, w_{i+1}) is either an edge in Y_t, or the subpath is fully contained in $Y \cap E(G_{t'})$ or $Y \cap E(G_{t''})$, and uses internal vertices in W. In the first case, $(w_i, w_{i+1}) \in Y_t$, while in the remaining cases, (w_i, w_{i+1}) is in $\Gamma_{t'}$ or $\Gamma_{t''}$, by the induction hypothesis. We conclude that, since $w_i \in Z_t = W \cap X_t$, for $i \in [l-1]$, then (u, v) is in

$$\mathrm{tc}^*_{Z_t}\left(\Gamma^*_{t'} \cup \Gamma^*_{t''} \cup Y_t\right)\Big|_t$$

Equivalence for Δ^*_t: We now prove that both definitions for Δ^* are equivalent, by using induction on the depth of the nodes. For the node $\mathrm{root}(\mathcal{T})$, we have $E(G_{\mathrm{root}(\mathcal{T})}) = E(G)$, and thus the base case follows.

For the induction step, we remark that $\Delta^*_{p(t)}, \Gamma^*_t \subseteq \mathrm{tc}^*_W(Y)$ by the induction hypothesis together with the statement of the lemma for Γ^*. Therefore,

$$\mathrm{tc}^*_{Z_t}\left(\Delta^*_{p(t)} \cup \Gamma^*_t\right)\Big|_t \subseteq \mathrm{tc}^*_W(Y)\Big|_t$$

For the reverse inclusion, we fix a pair $(u,v) \in \mathrm{tc}^*_W(Y)\Big|_t$, and a path p that connects u to v in Y using internal vertices in W. Further, let $u = w_0, w_1, \ldots, w_\ell = v$ be all the vertices of p (in the correct order) that are also in X_t.

By Lemma A.2, (w_i, w_{i+1}) is either an edge in $Y(b)$, or the subpath p' of p connecting w_i and w_{i+1} is contained either in $Y \cap E(G_{t'})$, $Y \cap E(G_{t''})$ or $Y \cap (E \setminus E(G_t))$. For all but the last case, $(w_i, w_{i+1}) \in \Gamma^*_t$, by definition. In the remaining case, it must be that the vertices of p' are also contained in $X_{\mathcal{T}\setminus\mathcal{T}_b}$. Specifically, because the nodes whose bags contain a given vertex must form a connected component of $\mathcal{T}$, $w_i, w_{i+1} \in X_{p(t)}$, which implies $(w_i, w_{i+1}) \in \mathrm{tc}^*_W(Y)\Big|_{p(t)} = \Delta^*_{p(t)}$.

In any case, since $w_i \in Z_t = W \cap X_t$, for $i \in [l-1]$, we conclude that every pair (w_i, w_{i+1}) and thus (u,v), are contained in $\mathrm{tc}^*_{Z_t}\left(\Delta^*_{p(t)} \cup \Gamma^*_t\right)\Big|_t$. □

B Details of the DP for EC-SNDP

B.1 Correctness

The following two lemmas imply the correctness of our DP.

Lemma B.1. *Let $F \subseteq E(G)$ be a feasible solution. Then, for every node t, there is a profile Ψ_t for t such that (t, Ψ_t) is valid; for any node t with children t', t'', then $\Psi \xleftrightarrow{Y_t} (\Psi', \Psi'')$, where $Y_t = F \cap E_t$. Furthermore, $c[\mathrm{root}(\mathcal{T}), \Psi_{\mathrm{root}(\mathcal{T})}] = c(F)$.*

Proof. We first observe that for each demand (γ_i, k_i), $i \in [h]$ the solution F can be partitioned into $F = \bigcup_{j\in[k]} F^i_j$ such that the partition F^i_j contains a path connecting γ_i to the root r. Note that, if $k_i < k$, there might not be any path in the partitions $F_{k_i+1}, \ldots, F_k$.

Let t be a node in $\mathcal{T}$. We define Ψ_t as follows. We define the pair (omitting the script of t for convenience) $(\vec{\Gamma}^i(t), \vec{\Delta}^i(t))$ for all $i \in [h]$ as follows.

$$\Gamma^i_j(t) = \mathrm{tc}(F^i_j \cap E(\mathcal{T}_t))\Big|_t$$
$$\Delta^i_j(t) = \mathrm{tc}(F^i_j)\Big|_t$$

We now define $\Psi_t = \{(\vec{\Gamma}^i(t), \vec{\Delta}^i(t)) : i \in [h]\}$.

We first show that any cell (t, Ψ_t) is valid. For any leaf node t, $E_t = \emptyset$ and hence $\Gamma^i_j(t) = \emptyset$, $i \in [h]$, $j \in [k]$. For the node $\mathrm{root}(\mathcal{T})$, $E(\mathcal{T}_t) = E$ and hence $\Gamma^i_j(t) = \Delta^i_j(t)$, $i \in [h]$, $j \in [k]$. Finally, since

$\mathcal{P}_j^i$ connects r to γ_i for all $i \in [h]$, $j \in [k_i]$, then $(r, \gamma_i) \in \Delta_j^i(t_{\gamma_i})$. We conclude that any cell (t, Ψ_t) defined as above is marked valid.

Finally we define, for any node t, a set of edges $Y(t) \subseteq E_t$ along with a suitable partition of $Y(t)$. Let $Y(t) = F \cap E_t$ and $Y_j^i(t) = F_j^i \cap E_t$. The subsets $Y_j^i(t)$ indeed form a partition owing to the disjointness of the sets F_j^i, for all $j \in [k_i]$ and a fixed $i \in [h]$.

It is clear from the above definitions of Γ_j^i and Δ_j^i that they satisfy the global connectivity conditions for Y_j^i. Applying Lemma 3.1, the local conditions must also be satisfied for the node t along with its children t', t'' and $Y(t)$. This gives us $\Psi \overset{Y(t)}{\longleftrightarrow} (\Psi', \Psi'')$ and we are done.

Notice that the $c[\text{root}(\mathcal{T}), \Psi_{\text{root}(\mathcal{T})}] = \sum_{t \in V(\mathcal{T})} c(Y(t)) = c(F)$. □

We remark that the DP uses exactly one cell per node in any valid solution.

Lemma B.2. *Let $\mathcal{C} = \{(t, \Psi_t) : t \in V(\mathcal{T})\}$ be the set of cells selected by the dynamic programming solution. Then there exists a set of edges $F \subseteq E$ such that for each demand (γ_i, k_i), $i \in [h]$, there exist k_i edge-disjoint paths connecting r to γ_i. Furthermore, $c(F) = c[\text{root}(\mathcal{T}), \Psi_{\text{root}(\mathcal{T})}]$.*

Proof. First we define the set F. Consider the cells in $\mathcal{C}$. Since each cell (t, Ψ_t) is picked by the DP, there must exist a set of edges $Y_t \subseteq E_t$ such that for some pair of children cells $\{(t', \Psi_{t'}), (t'', \Psi_{t''})\} \in \mathcal{C}$, $\Psi_t \overset{Y_t}{\longleftrightarrow} (\Psi_{t'}, \Psi_{t''})$. Define $F = \bigcup_{(t, \Psi_t) \in \mathcal{C}} Y_t$. We prove that for any demand (γ_i, k_i), $i \in [h]$, there exist edge-disjoint paths $\mathcal{P}_j^i$, $j \in [k_i]$ in F that connect r to γ_i.

For a demand vertex γ_i, $i \in [h]$, consider the node $t^\star = t_{\gamma_i}$ and the cell $(t^\star, \Psi_{t^\star}) \in \mathcal{C}$. Since this cell is valid, there exists a connection profile $(\vec{\Gamma}, \vec{\Delta}) \in \Psi_{t^\star}$ such that $(r, \gamma_i) \in \Delta_j$ for all $j \in [k_i]$.

We now suitably define connection profiles $(\vec{\Gamma}^t, \vec{\Delta}^t)$ and sets of edges $Y_{t,j}$ for every other node $t \in V(\mathcal{T})$, and prove that these elements satisfy the local connectivity property of Lemma 3.1. We explicitly describe the definition for the children nodes t' and t'' of $t^\star$. The definition for every other node in the subtree $\mathcal{T}_t$ can be carried out in a similar recursive fashion.

By definition of consistent DP cells, there exists $(\vec{\Gamma}', \vec{\Delta}') \in \Psi_{t'}$, $(\vec{\Gamma}'', \vec{\Delta}'') \in \Psi_{t''}$ and a partition $Y_{t^\star} = \bigcup_{j \in [k]} Y_{t^\star, j}$ such that, for all $j \in [h]$, the triplet $(\Gamma_j, \Delta_j), (\Gamma'_j, \Delta'_j), (\Gamma''_j, \Delta''_j)$ satisfies the local connectivity conditions for $Y_{t^\star, j}$. We take $(\vec{\Gamma}^{t'}, \vec{\Delta}^{t'}) = (\vec{\Gamma}', \vec{\Delta}')$, $(\vec{\Gamma}^{t''}, \vec{\Delta}^{t''}) = (\vec{\Gamma}'', \vec{\Delta}'')$.

A similar recursive definition works for all nodes in $\mathcal{T} \setminus \mathcal{T}_t$, starting with $(\vec{\Gamma}, \vec{\Delta})$ and defining a suitable connection profile in $p(t)$.

We now apply the equivalence from Lemma 3.1 to conclude that the global connectivity definition is satisfied by (Γ_j^t, Δ_j^t) and edge set $Y_{t,j}$ for any node $t \in V(\mathcal{T})$. As a consequence, there exist paths connecting r to γ_i in $\bigcup_{t \in V(\mathcal{T})} Y_{t,j}$, for $j \in [k_i]$. Since the sets $Y_{t,j}$ are disjoint for $j \in [k_i]$, we obtain the required k_i edge-disjoint paths in the sets $Y_j = \bigcup_{t \in V(\mathcal{T})} Y_{t,j}$. □

B.2 Running Time Analysis

To bound the running time of the DP algorithm, we start by proving a bound on the number of cells in the DP table (Lemma 4.1), and then show how this implies the running time of the algorithm.

Proof of Lemma 4.1.

The following observations are used to prove the lemma:

Observation B.3.

1. *The number of possible subsets of edges between w elements is 2^{w^2}.*
2. *The number of possible partitions of such a subset of edges is k^{w^2}.*
3. *The number of possible equivalence relations in a set of w elements is w^w.*

Let $t \in \mathcal{T}$. We know that $\Psi \subseteq \mathcal{X}_t \times \mathcal{X}_t$, whose size can be up to 2^{kw^2}. However, it is sufficient to consider equivalence relations in the node t, as we always take the transitive closure in definitions. Therefore, there are at most w^w possibilities for each Γ_j, Δ_j (Observation B.3) and at most $2^{w^{2wk}}$ possibilities for Ψ. □

Note that the algorithm itself does one of the two possible options for each cell: it either initializes itself, for which it needs to check every element of Ψ, taking time $O(w^{2wk}kw)$; or it computes the value based on children cells, for which it enumerates all sets Ψ', Ψ'' and checks them for consistency. The number of such sets to check is again $2^{w^{2wk}}$, and checking each triple of sets takes time polynomial in w^{wk} and k^{w^2}. In sum, the algorithm takes time $O(n \exp(w^{O(wk)}))$.

C Details on the Algorithms for Restricted Group SNDP

In this section, we present Lemma C.1 and Lemma C.2, which prove the correctness of the reduction presented in Section 5 and complete the proof of Theorem 5.1.

Lemma C.1. *For every solution $F \subseteq E(G)$, there is a valid tree $\widetilde{T} \subseteq \widetilde{\mathcal{T}}$ with the same cost and that connects the same groups that F k_i-connects, that is, if F contains k_i edge-disjoint paths to a vertex $\gamma_i \in S_i$, then $\widetilde{T}$ connects the root to some node $\tilde{t} \in \widetilde{S}_i$.*

Proof. Let $\{\gamma_i \in S_i\}_{i\in[h']}$, $h' \leq h$, be the group vertices that are k_i-connected by F (w.l.o.g. F the first h' groups). For every $i \in [h']$, we partition the solution F into k_i subsets that connect r to γ_i, i. e. we partition F into $\{F_{ij}\}_{j\in[k_i]}$, such that each F_{ij} contains a r-γ_i-path. We also define Ψ_t and $(\vec{\Gamma}^i(t), \vec{\Delta}^i(t)) \in \Psi_t$ for all $t \in V(\mathcal{T})$ as is done in the proof of Lemma B.1.

By the proof of Lemma B.1, $(\Psi_{t'}, \Psi_{t''}) \overset{F\cap E_t}{\longleftrightarrow} \Psi_t$. Therefore, we can build a tree $\widetilde{T}$ by picking one copy of each of the nodes $\tilde{t}[t, \Psi_t]$ and connecting nodes $\tilde{t}_c[t, \Psi_t, \Psi_{t'}, \Psi_{t''}, F \cap E_t]$, such that $\widetilde{T}$ is connected.

Furthermore, for the topmost node t_{γ_i} containing γ_i, we have that $\tilde{t}[t_{\gamma_i}, \Psi_{t_{\gamma_i}}] \in \widetilde{S}_i$, since F_{ij} connects r and γ_i, which implies that $(r, \gamma_i) \in \Delta^i_j(t_{\gamma_i})$, for $j \in [k_i]$. Therefore, if F k_i-connects a group S_i, $\widetilde{T}$ contains a node of $\widetilde{S}_i$.

The proof that F and $\widetilde{T}$ have the same cost follows from the proof of Lemma B.1. □

Lemma C.2. *For every valid tree $\widetilde{T} \subseteq \widetilde{\mathcal{T}}$, there is a solution $F \subseteq E(G)$ with the same cost and which k_i-connects a group S_i if $\widetilde{T}$ connects $\widetilde{S}_i$, that is, if $\widetilde{T}$ connects the root to some node $\tilde{t} \in \widetilde{S}_i$, then F contains k_i edge-disjoint paths to a vertex $\gamma_i \in S_i$.*

Proof. Let $\{\widetilde{S}_i\}_{i\in[h']}$, $h' \leq h$ be the groups connected by $\widetilde{T}$ (w.l.o.g.). By the definition of $\widetilde{S}_i$, there must be some $\gamma_i \in S_i$ and node $\tilde{t}[t_{\gamma_i}, \Psi_{t_{\gamma_i}}] \in \widetilde{S}_i \cap V(\widetilde{T})$ for every $i \in [h']$.

By the definition of valid tree, $\widetilde{T}$ contains exactly one node $\tilde{t}[t, \Psi_t]$ for each $t \in V(\mathcal{T})$, as well as exactly one node $\tilde{t}_c[t, \Psi_t, \Psi_{t'}, \Psi_{t''}, Y_t]$ for each non-leaf node $t \in V(\mathcal{T})$ such that

$$\Psi_t \xleftrightarrow{Y_t} (\Psi_{t'}, \Psi_{t''}).$$

The above conditions suffice to apply the proof of Lemma B.2 (with demands $\{(\gamma_i, k_i)\}_{i\in[h']}$); thus, we can obtain a solution F such that for every $i \in [h']$, F contains k_i edge-disjoint paths from r to γ_i. Furthermore, F has the same cost as $\widetilde{T}$. □

D Algorithms for VC-SNDP

In this section, we focus on vertex connectivity problems, specifically, Rooted VC-SNDP with multiple roots, where there is a subset $R \subseteq V$ such that the connectivity requirement $k(u, v) > 0$ only if $u \in R$.

We argue that any instance of Subset k-VC can be transformed into that of Rooted VC-SNDP with k roots as follows: Let T be a terminal set for k-subset vertex connectivity. Let $R \subseteq T$ be arbitrary subset of terminals with $|R| = k$. We specify the connectivity $k(r, v) = k$ for all $r \in R$ and $v \in T$. It is easy to verify that this instance is equivalent to the original instance. In the subsequent discussion, we focus on Rooted VC-SNDP with at most k roots. We start by adding, for each $r \in R$, vertex r into every node t. This increases the width of the decomposition by an additive factor of at most k.

The organization in this section is as follows. We start by extending the framework of Section 3 to vertex-connectivity in Appendix E. In Appendix E.1, we explain how the DP table is setup and the main ideas for the vertex-connectivity version of the problem. In Appendix E.2, we present the DP algorithm, and how to compute the value of each cell. Finally, in Appendix E.3 we show that the algorithm finds an optimal solution to the VC-SNDP problem. Throughout this section, we will use the notation defined in Section 4, when applicable.

E Global ⇔ Local Checking for Vertex Connectivity

When dealing with vertex connectivity, we want to find paths that have disjoint internal vertices, but which all share the same source and sink. Therefore, we must introduce some changes to our connection sets and to the way they are defined.

The most important change is the use of a function tc*, which is a modified version of transitive

closure, introduced in Appendix A. We recall its definition here:

$$\mathrm{tc}^*_Z(S) = \Big\{(u,v) \;\Big|\; \exists w_1,\ldots,w_\ell \in Z, \forall i \in [\ell-1], (u,w_1),(w_\ell,v),(w_i,w_{i+1}) \in S\Big\}$$

Our previous notion of pair of sets (Γ,Δ) is now extended with a subset $Z \subseteq X_t$, which represents the vertices that can be used as internal nodes in a bag. We denote the new triple (Z,Γ^*,Δ^*). Let $W_t \subseteq X_t \setminus X_{p(t)}$ for every $t \in V(\mathcal{T})$, $W = \bigcup_{t\in V(\mathcal{T})} W_t$, and a triple $(Z,\Gamma^*_t,\Delta^*_t)$ for every $t \in V(\mathcal{T})$.

Given W, we say that the triples $(Z_t,\Gamma^*_t,\Delta^*_t)$ satisfy the *local* (resp. *global*) *connectivity definition* if, for every node $t \in V(\mathcal{T})$,

Local

$$Z_t := \begin{cases} W_t & \text{if } t = \mathrm{root}(\mathcal{T}) \\ \Big(Z_{p(t)} \cup W_t\Big) \cap V_t & \text{otherwise} \end{cases}$$

$$\Gamma^*_t := \begin{cases} \emptyset & t \text{ is a leaf node} \\ \mathrm{tc}^*_{Z_t}(\Gamma^*_{t'} \cup \Gamma^*_{t''} \cup E_t)\Big|_t & \text{otherwise} \end{cases}$$

$$\Delta^*_t := \begin{cases} \Gamma^*_t & \text{if } t = \mathrm{root}(\mathcal{T}) \\ \mathrm{tc}^*_{Z_t}(\Delta^*_{p(t)} \cup \Gamma^*_t)\Big|_t & \text{otherwise} \end{cases}$$

Global

$$Z_t := W \cap X_t$$
$$\Gamma^*_t := \mathrm{tc}^*_W\left(E(G_t)\right)\Big|_t$$
$$\Delta^*_t := \mathrm{tc}^*_W\left(E(G)\right)\Big|_t$$

As in Section 3, we prove that the the local and global connectivity definitions are equivalent (Lemma E.1).

Lemma E.1. *Let $W_t \subseteq X_t \setminus X_{p(t)}$ be a subset of vertices and $(Z,\Gamma^*_t,\Delta^*_t)$ a triple for every $t \in V(\mathcal{T})$. Then, the triples $(Z_t,\Gamma^*_t,\Delta^*_t)$ satisfy the local connectivity definition iff they satisfy the global one.*

The proof of the lemma follows from Lemma A.1, by setting $Y_t = E_t$.

The notions of local and global connectivity in Lemma E.1 are also used to design algorithms for edge-connectivity with node-costs. However, we do not need the modified definition of transitive closure $\mathrm{tc}^*(\cdot)$ and use the $\mathrm{tc}(\cdot)$ operator instead. We omit the full definition to avoid repetition. We remark that, even though we can use $\mathrm{tc}^*(\cdot)$, even for edge-connectivity, we can achieve a better runtime using $\mathrm{tc}(\cdot)$, as is shown in Section 4.

E.1 Profiles

As before, let $\mathcal{X}_t$ be the set of all connection profiles for t. By analogy, we say a *vertex profile* for t is a k-tuple $\vec{Z} = (Z_1, Z_2, \ldots, Z_k)$ is such that $Z_i \subseteq X_t$. Additionally, let $\mathcal{V}_t$ denote the set of all vertex profiles for t.

The profile Ψ of node t is now a collection of triples $(\vec{Z},\vec{\Gamma}^*,\vec{\Delta}^*)$, where $\vec{Z}$ is a vertex profile and $\vec{\Gamma}^*$, $\vec{\Delta}^*$ are connection profiles, i.e., $\Psi \subseteq \mathcal{V}_t \times \mathcal{X}_t \times \mathcal{X}_t$.

A partial solution $H \subseteq V(G_t)$ is said to be *consistent* with profile Ψ for t if, for all $(\vec{Z},\vec{\Gamma}^*,\vec{\Delta}^*) \in \Psi$,

- For each $(u,v) \in \Gamma_i^*$ and $(a,b) \in \Gamma_j^*$ for $i \neq j$, there are paths $P_{uv}, P_{ab} \subseteq H$ connecting the respective vertices in G_t such that the internal vertices of P_{uv} and P_{ab} are disjoint. All the internal vertices in P_{uv}, P_{ab} which are in X_t are contained in Z_i, Z_j, respectively.
- There is a global solution $H' \supseteq H$ such that, for each $(u,v) \in \Delta_i$ and $(a,b) \in \Delta_j$ for $i \neq j$, there are paths $Q_{uv}, Q_{ab} \subseteq H'$ connecting the respective vertices in G such that the internal vertices of Q_{uv} and Q_{ab} are disjoint. All the internal vertices in Q_{uv}, Q_{ab} which are in X_t are contained in Z_i, Z_j, respectively.

In other words, a solution consistent with a profile must "implement" all connectivity requirements by $\vec{\Gamma}$ and must be extensible to satisfy $\vec{\Delta}$. Furthermore, the vertices used by the solution in X_t are given by $\vec{Z}$.

The DP table has a cell $c[t, \Psi]$ for each node $t \in V(\mathcal{T})$ and each profile Ψ for t. This cell tentatively stores the optimal cost of a solution consistent with the profile.

E.2 The DP

Valid cells: We mark all the cells that do not correspond to valid solutions of the problem as invalid (for example, by setting $c[t, \Psi] = \infty$ for invalid cells). In particular, the following cells are invalid:

- Any leaf that has any connectivity requirements is invalid. That is, we set $c[t, \Psi] = 0$ if $\Psi \subseteq \mathcal{V}_t \times \{(\emptyset, \ldots, \emptyset)\} \times \mathcal{X}_t$; otherwise, $c[t, \Psi] = \infty$.
- Any cell that cannot be extended into a feasible solution is invalid. The cell is valid if, for every $r \in R$, there is a triple $(\vec{Z}, \vec{\Gamma}, \vec{\Delta}) \in \Psi$ such that $(r, v) \in \Delta_j$ for all $j \leq k(r,v)$ and $r, v \in \bigcup_{j \in [k]} Z_i$.

  Otherwise, either r or v are not in the solution, or they are not connected by $k(r,v)$ vertex-disjoint paths, and hence the cell is invalid.
- The root node $\mathrm{root}(\mathcal{T})$ together with profile Ψ is an invalid cell if there is a triple $(\vec{Z}, \vec{\Gamma}, \vec{\Delta}) \in \Psi$ such that $\Gamma_j \neq \Delta_j$ for some j.

DP Computation: For all other cells, we compute their values from the values of the children. Let t be a node with left-child t' and right-child t''. Let Ψ, Ψ', Ψ'' be their profiles, respectively, and $W \subseteq X_t \setminus X_{p(t)}$.

We say that Ψ is consistent with (Ψ', Ψ'') via W (abbreviated by $\Psi \xleftrightarrow{W} (\Psi', \Psi'')$) if the following conditions are satisfied. For each triple $(\vec{Z}, \vec{\Gamma}, \vec{\Delta}) \in \Psi$, there are $(\vec{Z}', \vec{\Gamma}', \vec{\Delta}') \in \Psi'$ and $(\vec{Z}'', \vec{\Gamma}'', \vec{\Delta}'') \in \Psi''$, together with a partition of W into $W_1 \cup W_2 \cup \ldots \cup W_k$ such that, for all $j \in [k]$,

- $\Gamma_j = \mathrm{tc}^*_{Z_t}(\Gamma'_j \cup \Gamma''_j \cup E_t)\Big|_t$
- $\Delta'_j = \mathrm{tc}^*_{Z_{t'}}(\Delta_j \cup \Gamma'_j)\Big|_{t'}$

- $\Delta''_j = \mathrm{tc}^*_{Z_{t''}}(\Delta_j \cup \Gamma''_j)\Big|_{t''}$
- $Z_j \cap (X_t \setminus X_{p(t)}) = W_j$
- $Z'_j \cap X_t = Z_j \cap X_{t'}$
- $Z''_j \cap X_t = Z_j \cap X_{t''}$

Similarly, for any triple $(\vec{Z}', \vec{\Gamma}', \vec{\Delta}') \in \Psi'$ (resp., $(\vec{Z}'', \vec{\Gamma}'', \vec{\Delta}'') \in \Psi''$), there are $(\vec{Z}'', \vec{\Gamma}'', \vec{\Delta}'') \in \Psi''$ (resp., $(\vec{Z}', \vec{\Gamma}', \vec{\Delta}') \in \Psi'$), plus $(\vec{Z}, \vec{\Gamma}, \vec{\Delta}) \in \Psi$ and some partition of W satisfying similar conditions as above.

We remark that the condition $Z_t = (Z_{p(t)} \cup W_t) \cap X_t$ in the local connectivity definition of Lemma E.1, can be equivalently written as $Z_t \cap X_{p(t)} = Z_{p(t)} \cap X_t$ and $Z_t \cap (X_t \setminus X_{p(t)}) = W_t$ (since W_t and $Z_{p(t)}$ are always disjoint). We choose to use the second formulation for convenience.

Then the value of $c[t, \Psi]$ can be defined recursively among valid cells:

$$c[t, \Psi] = \min_{\Psi \xleftrightarrow{W} (\Psi', \Psi'')} (c[t', \Psi'] + c[t'', \Psi''] + c(W))$$

Solution: The solution can be computed by

$$\min \left\{ c[\mathrm{root}(\mathcal{T}), \Psi] \mid (\forall (\vec{Z}, \vec{\Gamma}, \vec{\Delta}) \in \Psi)(\forall j \in [k]) \Gamma_j = \Delta_j \right\}$$

E.3 Correctness of the DP

The following two lemmas imply correctness.

Lemma E.2. *Let $H \subseteq V(G)$ be a feasible solution. Then, for every node t, there is a profile Ψ_t for t such that (t, Ψ_t) is valid; for any node t with children t', t'', then $\Psi \xleftrightarrow{W_t} (\Psi', \Psi'')$, where $W_t = H \cap (X_t \setminus X_{p(t)})$. Furthermore, $c[\mathrm{root}(\mathcal{T}), \Psi_{\mathrm{root}(\mathcal{T})}] = c(H)$.*

Proof. We fix $r \in R$, $v \in T$ with $k(r, v) > 0$. Observe that $r, v \in H$, and the solution H can be partitioned into $H = \{H_j\}_{j \in [k]}$ such that for $j \leq k(r, v)$, there is a path $\mathcal{P}_j$ between r and v whose internal vertices are in H_j. Note that, if $k(r, v) < k$, then H_j is irrelevant, for $j > k(r, v)$, and might be empty.

For any node t, we define Ψ_t as follows. The triple $(\vec{Z}(t), \vec{\Gamma}^*(t), \vec{\Delta}^*(t))$ is defined as:

$$\begin{aligned} Z_j(t) &:= H_j \cap X_t \\ {\Gamma^*}_j(t) &:= \mathrm{tc}^*_{H_j}(E(G_t))\Big|_t \\ {\Delta^*}_j(t) &:= \mathrm{tc}^*_{H_j}(E(G))\Big|_t \end{aligned}$$

We add to Ψ_t the triples $(\vec{Z}(t), \vec{\Gamma}^*(t), \vec{\Gamma}^*(t))$ for all $r \in R$, $v \in T$.

We first show that any cell (t, Ψ_t) is valid. We again fix $r \in R$, $v \in T$ and take the corresponding triples $(\vec{Z}(t), \vec{\Gamma}^*(t), \vec{\Delta}^*(t)) \in \Psi_t$ and partition $\{H_j\}_{j\in[k]}$.

For every leaf node t, $E_t = \emptyset$ and hence ${\Gamma^*}_j(t) = \emptyset$. For the root node, $t = \text{root}(\mathcal{T})$, $E(\mathcal{T}_t) = E$ and hence ${\Gamma^*}_j(t) = {\Delta^*}_j(t)$. Finally, for $j \in [k(r, v)$, $\mathcal{P}_j$ connects r to γ_i using internal vertices in H_j, so $(r, v) \in {\Delta^*}_j(t_v)$. Since, $r, v \in \bigcup_{j\in[k]} Z_j(t_v)$, we conclude that the cells (t, Ψ_t) defined above are marked valid.

For each node $t \in V(\mathcal{T})$, we additionally define $W_j(t) = H_j \cap (X_t \setminus X_{p(t)})$. Observe that the $W_j(t), j = 1, 2, \ldots, k_i$ naturally induce a partition on W_t. It is clear from the definitions that $Z_j(t)$, ${\Gamma^*}_j(t)$, ${\Delta^*}_j(t)$ satisfy the global connectivity definitions for vertex sets $W_j(t)$. Therefore, we can apply Lemma E.1, for every $j \in [k]$ to conclude that

$$\Psi_t \xleftrightarrow{W_t} (\Psi_{t'}, \Psi_{t''})$$

for every node $t \in V(\mathcal{T})$ with children t', t'', where $W_t = H \cap (X_t \setminus X_{p(t)})$.

Notice that the $c[\text{root}(\mathcal{T}), \Psi_{\text{root}(\mathcal{T})}] = \sum_{t\in V(\mathcal{T})} c(W_t) = c(H)$. □

We remark that every DP solution uses exactly one cell per node $t \in V(\mathcal{T})$.

Lemma E.3. *Let $\{(t, \Psi_t)\}_{t\in V(\mathcal{T})}$ be the cells selected in a dynamic programming solution. There exists a solution $H \subseteq V$ with cost $c(H) = c[\text{root}(\mathcal{T}), \Psi_{\text{root}(\mathcal{T})}]$ such that for every demand (γ_i, k_i), $i \in [h]$, H contains k_i vertex-disjoint paths from r to γ_i.*

Proof. Let us start by defining the solution H. For every non-leaf $t \in V(\mathcal{T})$ with children t', t'', there must be some set W_t such that

$$\Psi_t \xleftrightarrow{W_t} (\Psi_{t'}, \Psi_{t''})$$

We take $H = \bigcup_{t\in V(\mathcal{T})} W_t$ as our solution.

Now, fix $r \in R$, $v \in V$. We will prove that H contains $k(r, v)$ vertex-disjoint paths from r to v. We now define, for each node $t \in V(\mathcal{T})$, the sets $(\vec{Z}(t), \vec{\Gamma}^*(t), \vec{\Delta}^*(t)) \in \Psi_t$, as well as partition $\{W_j(t)\}_{j\in[k]}$ of the vertices in W_t, as is done in the proof of Lemma B.2.

Now, since for every $j \in [k]$, $Z_j(t)$, $\Gamma^*_j(t)$, $\Delta^*_j(t)$ satisfy the local connectivity constraints, we can apply Lemma E.1, which implies that $Z_j(t)$, $\Gamma^*_j(t)$, $\Delta^*_j(t)$ satisfy the global connectivity constraints as well.

Since all cells in (t, Ψ_t) are valid, we know that $(r, v) \in \Delta^*_j(t)$ for $j \in [k(r, v)]$ and $r, v \in \bigcup_{j\in[k]} Z_j(t)$. Therefore, $H_j = \bigcup_{t\in V(\mathcal{T})} H_j(t)$ contains the internal nodes of a path between r and v, and $r, v \in H$. We conclude that H contains $k(r, v)$ vertex-disjoint paths between r and v. □

E.4 Running Time Analysis

The running time for the presented algorithm follows closely that for the edge-connectivity version, with the exception that Γ^* and Δ^* are, in general, no longer equivalence relations, and there are now $2^{(w+k)^2}$ possibilities for each such set.

We conclude that the running time of the algorithm is $O(n \cdot \exp(\exp(O((w+k)^2 k))))$. We also remark that a very similar idea for the DP works for the edge-connectivity version of the problem with node-costs. However, we can use the definition of tc instead of tc* in order to define the DP cells and hence the runtime for that case is $O(n \cdot \exp(\text{poly}(w^{wk}))$.

F Hardness of Restricted Group SNDP on General Graphs

In this section, we sketch the proof of the approximation hardness for the vertex-weighted Restricted Group SNDP on general graphs. The reduction is from *Min-k-CSP*.

In *Max-k-CSP*, we are given a set of n variables $x_1, \ldots, x_n$ over the domain $[N]$ and a set of m constraints (which are functions) $C_1, \ldots, C_h$ such that each C_j depends on (exactly) k variables. The goal in the Max-k-CSP is to find an assignment to variables that maximizes the number of constraints satisfied. In the minimization version, say *Min-k-CSP*, we are allowed to assign multiple values (or *labels*) to each variable to guarantee that each constraint can be satisfied by some of the assignments while minimizing the total number of labels. To be specific the assignment is a function $\sigma : \{x_1, \ldots, x_n\} \to 2^{[N]}$, and we wish to find an assignment such that, for every constraint C_j depending on variables $x_{i_1}, \ldots, x_{i_k}$, there exist labels $a_1 \in \sigma(x_{i_1}), \ldots, a_k \in \sigma(x_{i_k})$ such that $C_j(a_1, \ldots, a_n)$ evaluates to *true* for all $j \in [m]$. It is know that if the hardness of Max-k-CSP is $\alpha(n)$, then the hardness of the Min-k-CSP is $\Omega(\alpha(n)^{1/k})$ (see, e.g., [28]). The special case of $k = 2$ is called the *Label-Cover* problem.

We reduce from Min-k-CSP to Restricted Group SNDP by representing the assignments of variables by positive weight vertices and representing accepting configurations of each constraints by a group of vertices. To be precise, we first construct a graph $G = (V, E)$ such that

$$\begin{aligned} V = {} & \{r\} \cup \{(x_i, a) : i \in [n], a \in [N]\} \cup \\ & \{(j, a_1, \ldots, a_k) : j \in [h], a_1, \ldots, a_k \in [N] \text{ and } C_j(a_1, \ldots, a_k) = \text{true}\} \\ E = {} & \{r(x_i, a) : i \in [n], a \in [N]\} \cup \\ & \{(x_i, a_i)(j, a_1, \ldots, a_k) : (x_i, a_i), (j, a_1, \ldots, a_k) \in V \text{ and} \\ & \quad C_j \text{ depends on } x_i \text{ at the } \ell\text{-th argument, and } a_\ell = a\} \end{aligned}$$

We set the cost of each vertex of the form (x_i, a) to be one and set cost of other vertices to be zero. We define r to be the root vertex and, for each $j \in [h]$, we define a group $S_j = \{(j, a_1, \ldots, a_k) \in V\}$. We set the connectivity demands $k_j = k$ for all $j \in [h]$. This finishes the construction.

Observe that every vertex that belongs to some group S_j has degree exactly k. Thus, the only way to have k-edge disjoint paths from some vertex in S_j to the root r is to choose all of its neighbors in $\{(x_i, a) : C_j \text{ depends on } x_i, a \in [N]\}$. It is not hard to see that vertices in the latter set corresponding to an assignment to variables in the Min-k-CSP instance and that some vertex in S_j has k neighbors if and only if the chosen vertices correspond to an assignment that satisfies the constraint C_j. Moreover, one may verify that a vertex $v \in S_j$ has k neighbors in an induced subgraph $H \subseteq G$ if and only if v has k edge-disjoint paths connecting to the root r. Consequently, this gives a reduction from Min-k-CSP to the vertex-weighted Restricted Group SNDP. The approximation hardness then follows from the hardness of Min-k-CSP, which is $2^{\log^{1-\epsilon} n}$

under $\mathsf{NP} \not\subseteq \mathsf{DTIME}(n^{\mathrm{polylog(n)}})$, and $n^{\delta/k}$ for some constant $0 < \delta < 1$ under the Sliding Scale Conjecture [2].