The Connected Critical Node Problem

The Connected Critical Node Problem

Pierre Hosteins
University Gustave Eiffel, COSYS, ESTAS,
F-59650 Villeneuve d'Ascq, Lille, France
e-mail: pierre.hosteins@ifsttar.fr
Rosario Scatamacchia
Politecnico di Torino,
Dipartimento di Ingegneria Gestionale e della Produzione,
Corso Duca degli Abruzzi 24 - 10129 Torino, Italy
Andrea Grosso, Roberto Aringhieri
Dipartimento di Informatica,
Università degli Studi di Torino,
Corso Svizzera 185, 10149 Torino, Italy

October 7, 2022

Abstract

The Critical Node Problem is a well-known optimization problem that aims to find the subset of nodes in a graph whose removal impacts the graph connectivity as much as possible according to a specific connectivity measure. In this work, we study a new version of the Critical Node Problem, which we call the *Connected Critical Node Problem*, where the set of the removed nodes has to form a connected subgraph. We consider three connectivity measures and provide complexity results and solution approaches for general graphs and specific classes of graphs such as graphs with bounded treewidth, trees and series-parallel graphs. We consider the Connected Critical Node Problem where the pairwise connectivity (related to the number of pairs of vertices still connected in the graph) is minimized. We prove that this problem is strongly NP-hard and inapproximable on general graphs and is polynomial-time solvable on graphs with bounded treewidth and with unit connection costs. Further, we prove the NP-hardness of the problem with arbitrary connection costs over trees and series-parallel graphs and derive dynamic programming algorithms. We extend our results to the problem variants that consider the minimization of the largest connected component and the maximization of the number of connected components (also called K-way Vertex Cut Problem). As side results, we provide new complexity results for the classic Critical Node Problem on series-parallel graphs.

Keywords: Critical Node Problem, Complexity, Dynamic Programming, Treewidth, trees, series-parallel graphs, K-way Vertex Cut Problem.

1 Introduction

The Critical Node Problem (CNP) belongs to the family of Network Interdiction Problems [42] and aims at maximally fragmenting a graph by deleting a subset of its nodes (and all incident edges on such nodes) according to a specific connectivity measure. Considerable attention has been centered on this problem in the literature due to its numerous applications, including the identification of key players in a social network [14], transportation networks vulnerability [28], power grid construction

and vulnerability [39], homeland security [15], telecommunications [4] or epidemic control [51] and immunisation strategies [9, 17, 45].

Different connectivity measures can be considered according to specific applications of interest and this choice typically leads to different optimal solution sets, as shown in [5, 40, 47]. The connectivity measures considered in the literature are often linked to the number of maximal connected components, their maximum cardinality or the overall number of node pairs connected by a path (the so called pairwise connectivity) [9, 10, 19, 40, 41]. At the current state of the art, many heuristic algorithms have been proposed for these problems [1, 5, 6, 37, 38, 45, 46]. We refer to [5, 47] for a comprehensive literature review of problems involving the main graph fragmentation metrics and to [33] for a recent and general survey of the CNP. Other alternative ways to quantify the fragmentation of a graph also exist, such as: the network's diameter [3], single/multiple-commodity maximum flow or the shortest path between given source-sink node pairs [26, 34, 36, 50] or functions of the shortest paths lengths between pairs of nodes [7, 48].

Two recent works consider versions of the CNP where the set of critical nodes has a specific structure. The work of [25] focus on finding a Critical Path, i.e. a set of critical nodes that creates a path in the graph between a source node and a sink node. The study of Walteros et al [49] considers sets of critical nodes that are either cliques or stars. The additional constraints on the set of deleted nodes make the CNP more tailored to specific applications of interest (such as, e.g., surveillance paths). Following this interesting trend, in this work we consider a variation of the CNP where the set of critical nodes has to be a *connected subgraph*. We denote this problem as the Connected CNP. The connectivity requirement can reflect many practical situations further motivating the study of such a CNP variant. In general terms, the Connected CNP finds applications when the nodes could be deleted only through diffusive processes that progressively remove the nodes in the graph or alternatively when the CNP is considered as a conquest problem where some resources in a graph (e.g. nodes) are taken by a player who wants to have connected resources for implementing appropriate defensive strategies. We mention below some specific application examples associated with the Connected CNP:

- The work of [25] focuses on surveillance applications where drones or patrols aim to intercept movements in a geographical area represented by a graph. Here the goal is to select paths for the drones constituted by the most critical nodes that connect a source node and a sink node in a graph. We can extend the analysis to situations where the surveillance patrol is allowed to move back and forward to explore relevant parts of a given area, defining a connected subgraph without a specific topology. Such an extension can be highly reasonable also for disaster relief applications.
- The study of influential processes in social networks has attracted a lot of attention in the recent years, see, e.g., [11, 20, 22, 35]. A social network can be typically represented by a graph where the nodes are people of the social network and each edge represents a friendship between two people. When one would like to spread an opinion or information in a social network, it is important to identify a community of people having a large impact on the network in terms of connectivity. At the same time, such a community must be convinced to spread the information. To this aim, in practice, it is usually easier to obtain the consensus of individuals if their friends have already been convinced, giving a more stable community for the spreading process (as also addressed in some recent versions of network domination problems). In these contexts, we can consider the Connected CNP so as to identify a connected and robust community that can partition the social network into small connected components which cannot communicate except through the critical set of selected people.

Let us formally recall the CNP. Consider an undirected graph $G = (V, E)$ with a set of vertices V

and a set of edges E. We denote by $n = |V|$ the number of vertices and by $m = |E|$ the number of edges. We denote by c_{ij} the *connection cost* between two nodes $i \in V$ and $j \in V$ in the graph, by k_i the *deletion cost* associated with the deletion of node $i \in V$, by S the set of deleted nodes (*critical set*) and by $G[V \setminus S]$ the graph induced by the set $V \setminus S$. Considering a budget K on the overall deletion costs, the classic CNP based on pairwise connectivity, thereafter called *Pairwise CNP*, can be formulated as follows:

$$\min_{S \subseteq V} \sum_{i<j} \{c_{ij} : i, j \text{ are connected in } G[V \setminus S]\} \tag{1}$$

$$\sum_{i \in S} k_i \leq K \tag{2}$$

This CNP variant was proven to be strongly NP-hard on general graphs in [9, 2]. A negative approximation result (under $P \neq NP$) is also given in [2], i.e., there does not exist a polynomial time approximation algorithm for the CNP.

The Pairwise CNP has also been studied over specially structured graphs. A class of graphs strongly investigated is the class of trees $\mathcal{T} = (V, E)$, often solved by Dynamic Programming (DP). In [18], it has been shown that the Pairwise CNP on trees with unit connection costs, namely when $c_{ij} = 1$ for all (i, j), is polynomially solvable (even with non-unit node deletion costs k_i) while the case with non-unit connection costs remains strongly NP-hard. Further results are provided in [2] where it is shown that the Pairwise CNP with unit connection costs is polynomially solvable on graphs with bounded tree-width. Polynomial and Pseudo-polynomial algorithms are provided in [7] for the Pairwise CNP over trees with non-unit but specific connection costs. Additional results on specially structured graphs with different types of objective functions have been provided in the works of [7, 32, 40]. A stochastic version of the problem over trees is studied in [27].

Another popular version of the CNP focuses on the minimization of the weight of the heaviest component (or its cardinality in case of unit node weights) instead of the pairwise connectivity, see, e.g., [2, 5, 10, 40, 41, 47]. This problem is sometimes called *MinMaxC* CNP. It is characterised by node deletion costs k_i as well as node weights c_i for all nodes $i \in V$. We define $c(H)$ as the total weight of the nodes inside a subset of nodes $H \subseteq V$ and $\text{comp}(G[V \setminus S])$ as the set of maximally connected components of the graph induced by $V \setminus S$. The objective function of the *MinMaxC* CNP is written as follows:

$$\min_{S \subseteq V} \max\{c(H) : H \in \text{comp}(G[V \setminus S])\} \tag{3}$$

As observed in [41], this version of the CNP is strongly NP-hard on general graphs with unit node weights.

Finally, the third version of the CNP considers the maximisation of the number of connected components after the deletion of a subset of nodes S. The objective function of this problem, denoted in [41] as the *MaxNum* CNP, is:

$$\max_{S \subseteq V} |\text{comp}(G[V \setminus S])| \tag{4}$$

This problem is also known in the literature as the K-Way Vertex Cut Problem (K-WVCP) and has been tackled, among others, in [2, 5, 12, 40, 41]. It is shown to be strongly NP-complete with unit deletion costs on general graphs in [41] by a reduction from the Independent Set Problem.

In this work, we introduce the connected version of the CNP and study the problem with the three objective functions indicated above. Our contributions are summarised as follows:

- We prove that the Connected CNP is strongly NP-hard, even on biconnected planar bipartite graphs, for all three objective functions. We also prove that the Connected CNP based on pairwise connectivity cannot be approximated by a constant factor in polynomial time.
- For the Connected Pairwise CNP with unit connection costs, we derive a DP algorithm over the *nice tree decomposition* of a graph and show that the algorithm is polynomial when the treewidth of the graph is bounded by a constant. We extend the proposed algorithm to the other objective functions as well.
- We prove that the Connected Pairwise CNP with arbitrary costs is NP-hard on trees and series-parallel graphs, and polynomial-time solvable on paths. We also provide efficient DP algorithms over trees for different input configurations and illustrate the corresponding differences in the complexity results between the Connected Pairwise CNP and the classic Pairwise CNP. We perform a similar analysis for the Connected *MinMaxC* CNP and the Connected *MaxNum* CNP.
- As side results, we prove the NP-hardness of the classic Pairwise CNP and the *MinMaxC* CNP over series-parallel graphs with arbitrary connection costs and arbitrary node weights, respectively, which were still open problems.

The paper is organised as follows. We first prove the strong NP-hardness and inapproximability of the Connected CNP over biconnected planar bipartite graphs in Section 2. We then describe a DP algorithm to solve some problem variants over nice tree decompositions of a general graph in Section 3 and show that the algorithm is polynomial when the treewidth of the graph is bounded by a constant. The proof of correctness of the algorithm is reported in Appendix A. We provide algorithms and complexity results over trees in Section 4 and series-parallel graphs in Section 5. For the sake of exposition, we only present the results for the Connected Pairwise CNP in the main body of the paper. Extensions of our results to Connected CNP versions based on the weight of the heaviest connected component and on the number of connected components are presented in Appendices B and C, respectively. Section 6 draws some conclusions.

2 NP-hardness and inapproximability on biconnected planar bipartite graphs

We derive complexity results for the decision version of the Connected CNP by considering the existing results for the Connected Minimum Vertex Cover problem. The decision version of the Connected CNP asks whether there exists a solution with an objective function not worse than a given threshold value. We prove the NP-completeness of the problem for the three objective functions considered in the paper in biconnected planar bipartite graphs of maximum degree 4.

> **Proposition 1:** The decision version of the Connected Pairwise CNP is strongly NP-complete in biconnected planar bipartite graphs of maximum degree 4, even with unit connection costs and unit deletion costs.

Proof. As already noticed in the literature [2], the Pairwise CNP with unit costs generalises the Minimum Vertex Cover problem on a given graph G. In fact, a solution of the Pairwise CNP has zero objective function (i.e., induces a disconnected subgraph) if and only if the same solution is a vertex cover of G with cardinality bounded by K ($< n$). Since the MinCVC, where the vertex cover constitutes a connected subgraph, has been shown to be NP-complete in biconnected planar

bipartite graphs of maximum degree 4 in [21], we can infer that the Connected Pairwise CNP with unit costs is also NP-complete in such graphs. Moreover, since the MinCVC tackled in [21] is not a number problem (all data parameters are equal to one), a pseudo-polynomial algorithm for the Connected Pairwise CNP would solve the MinCVC in polynomial time. Clearly this is not possible under $P \neq NP$, implying the NP-Completeness in the strong sense of the Connected Pairwise CNP with unit costs. □

The NP-completeness results for the Connected *MinMaxC* CNP and the Connected *MaxNum* CNP are reported in Appendices B and C. Employing an argument similar to the reasoning in [2], we also prove the following result:

Proposition 2: The Connected Pairwise CNP does not admit a polynomial time approximation algorithm with a bounded approximation ratio unless P = NP, even in biconnected planar bipartite graphs of maximum degree 4 with unit connection costs and unit deletion costs.

Proof. Consider an instance of the Connected Pairwise CNP over biconnected planar bipartite graphs of maximum degree 4, with unit connection and deletion costs and a budget capacity K. Deciding whether there exists a solution set S inducing a zero pairwise connectivity in $G[V \setminus S]$ corresponds to finding a connected vertex cover S in G with a size smaller than (or equal to) K, i.e. $|S| \leq K$. At the same time, deciding whether a connected vertex cover with a size not larger than K exists in biconnected planar bipartite graphs of maximum degree 4 is NP-complete. A polynomial time approximation algorithm with a bounded approximation ratio for the Connected Pairwise CNP would allow us to decide the connected vertex cover in polynomial time by just checking if its approximate solution has a strictly positive objective function. Clearly this is impossible under P $\neq$ NP. □

3 The Connected CNP over graphs with bounded treewidth

In this section we provide a DP algorithm over any nice tree decomposition of a graph. In particular, we extend the algorithmic framework proposed in [2] for the classic Pairwise CNP in order to handle the additional requirement of having a connected critical set. In our solution approach, we explicitly keep track of the number of connected components in the partial solutions of the critical set S during the steps of the algorithm and combine these solutions in polynomial time to eventually compute an optimal critical set with a single connected component. We start considering the Connected Pairwise CNP with unit connection costs ($c_{ij} = 1$ for each pair $i, j \in V$) and arbitrary deletion costs. Our results are then extended to the other objective functions considered in the previous section (see Appendices B and C).

3.1 Notations and definitions

Notation. Let $V(G)$ and $E(G)$ denote the node set and the edge set of an undirected graph G respectively; $\{u, v\}$ denotes an undirected edge, that is an element of $E(G)$; $N(v) = \{u : \{u, v\} \in E\}$ is the set of neighbours of a node v. For any graph H, let $\text{comp}(H)$ be the set of connected components of H; for any set $I \subset V(H)$ we denote by $G[I]$ the subgraph induced by I. We also denote the number of connected node pairs in graph H as

$$\#\text{conn}(H) = \sum \left\{ \binom{|V(\hat{H})|}{2} : \hat{H} \in \text{comp}(H) \right\}.$$

Tree decomposition. A *tree decomposition* of a connected graph G is a rooted tree T with node set $V(T)$ whose elements $X_i \in V(T)$ are called *bags* and are subsets of $V(G)$, satisfying the following properties:

(i) $\bigcup\{X : X \in V(T)\} = V(G)$;

(ii) $\{u, v\} \in E(G) \implies \exists X \in V(T) : u, v \in X$;

(iii) If P is the unique path in T linking two bags X_i, X_j

$$u \in X_i, u \in X_j \implies u \in X_k \text{ for all } X_k \in P.$$

In general, a graph can have many tree decompositions. The width of the tree decomposition T is $\max\{|X| : X \in V(T)\} - 1$; the *treewidth* of G is the minimum possible width of a tree decomposition of G. Although finding the treewidth of a general graph is an NP-hard problem [8], it is possible to construct (if it exists) a tree decomposition of width $\tau \leq \kappa$ in linear time, for any constant κ [13].

The union of a bag X_i with all its descendant bags in T is denoted V_i. A *nice* tree decomposition for graph G is a tree decomposition where, in addition to properties (i)-(iii) listed above, each bag falls in one of the following cases:

- a *leaf (or start) bag* X_i has no child and $|X_i| = 1$;
- a *join bag* X_i has two child bags $X_{i'}$,$X_{i''}$ such that $X_{i'} = X_{i''} = X_i$;
- a *forget bag* X_i has a single child X_j such that $X_i = X_j \setminus \{v\}$ for some $v \in V(G)$;
- an *introduce bag* X_i has a single child X_j such that $X_i = X_j \cup \{v\}$ for some $v \in V(G)$.

A nice tree decomposition with width τ can be obtained from any given tree decomposition with width τ in polynomial time, with an $\mathcal{O}(\tau n)$ number of bags, see, e.g., [30]. Finally, we recall two well-known and useful properties of a nice tree decomposition: (a) for an introduce with bag $X_i = X_j \cup \{v\}$ with child X_j, all neighbours of v in V_i are included in X_j; (b) for a join bag X_i with two children $X_{i'}$ and $X_{i''}$, the intersection between $V_{i'} \setminus X_i$ and $V_{i''} \setminus X_i$ is empty and there is no edge between a node in $V_{i'} \setminus X_i$ and a node in $V_{i''} \setminus X_i$.

3.2 Connected components configurations

In order to tackle the Connected Pairwise CNP with unit connection costs over a nice tree decomposition of a graph, we consider a specific structure from [2] called *connected component configuration* (CCC for short). Given a graph G and a set $\Sigma \subseteq V(G)$, the CCC of Σ in G is the set of pairs

$$\alpha = \{(\Sigma \cap V(H), |V(H) \setminus \Sigma|) : H \in \text{comp}(G), \Sigma \cap V(H) \neq \emptyset\}.$$

The number $|V(H) \setminus \Sigma|$ represents the number of nodes of a component H not appearing in Σ. We stress that only the components of G which have a *non-trivial intersection* with Σ are listed in the CCC. A collection $\alpha = \{(A_i, a_i)\}_{i=1}^{k}$ of subset-number pairs where the A_i's are a partition of $\Sigma \subseteq V(G)$ and $\sum_{i=1}^{k} a_i \leq |V(G)|$ is called a *potential* CCC, meaning that it may or may not be a CCC of Σ in G. We denote by $\Gamma(\Sigma, r)$ the set of all possible potential CCCs of Σ such that $\sum_{i=1}^{k} a_i \leq r$. We say that a potential CCC α is *realisable* if there exists a subset S such that $\Sigma \subseteq S \subseteq V(G)$ and α is the CCC of Σ in $G[S]$. In such a case, we call $G[S]$ a *realisation* of α.

We use the concept of CCC to capture the structure of the critical set S and its complement $V_i \setminus S$ *inside* a given bag X_i. In the dynamic programming algorithm presented in the next section, the set

Σ will represent the intersection of X_i with a partial critical set $S \subseteq V_i$. This allows us to keep track of the variation in the number of connected components in $G[S]$ and in the pairwise connectivity in $G[V_i \setminus S]$ when a node is added to or removed from a bag (for introduce and forget bags).

We will employ in the dynamic program the following operations on CCCs, taken from [2], to process forget, introduce and join bags, respectively. Let $k = |\alpha|$ be the number of pairs in a potential CCC α.

Restriction. Given $\alpha = \{(A_i, a_i)\}_{i=1}^{k} \in \Gamma(\Sigma, r)$ and $v \in \Sigma$, we define

$$\alpha - v = \{(A_i \setminus \{v\}, a_i + |A_i \cap \{v\}|) \colon i = 1, \ldots, k \text{ s.t. } |A_i \setminus \{v\}| > 0\}.$$

Hence, the operation $\alpha - v$ induces a potential CCC $\in \Gamma(\Sigma \setminus \{v\}, r)$ where the node v is removed from the corresponding set A_i in α. If v is the only node in the set, then the associated pair (A_i, a_i) is disregarded.

Extension. Given $\alpha = \{(A_i, a_i)\}_{i=1}^{k} \in \Gamma(\Sigma, r)$, $v \in \Sigma$ and $L = \{i : A_i \cap N(v) \neq \emptyset\}$, we define

$$\alpha + v = \{(A_i, a_i) : i = 1, \ldots, k \text{ s.t. } i \notin L\} \cup \left\{ \left(\{v\} \cup \bigcup_{i \in L} A_i, \sum_{i \in L} a_i \right) \right\}.$$

The operation $\alpha + v$ induces a potential CCC $\in \Gamma(\Sigma \cup \{v\}, r)$ where the node v is (possibly) merged with the sets A_i in α containing at least one neighbour of v.

Sum. For $\alpha' = \{(A'_i, a'_i)\}_{i=1}^{k'} \in \Gamma(\Sigma, r)$ and $\alpha'' = \{(A''_i, a''_i)\}_{i=1}^{k''} \in \Gamma(\Sigma, r)$, we define the *sum* CCC $\alpha = \alpha' + \alpha''$ as $\alpha = \{(A_i, a_i)\}_{i=1}^{k}$ such that

- u, v appear in the same A_ℓ if u, v appear in the same A'_i of α' or A''_i of α'';
- $a_\ell = \sum_{A'_i \subseteq A_\ell} a'_i + \sum_{A''_i \subseteq A_\ell} a''_i$.

The following lemmas allow us to consistently restrict, extend and sum CCCs over a nice tree decomposition.

Lemma 1 (Lemma 9 in [2])**.** *Consider a bag X_j in a nice tree decomposition T and $\Sigma \subset X_j$, and suppose there exists S such that $\Sigma \subseteq S \subseteq V_j \setminus (X_j \setminus \Sigma)$ and α is the CCC of Σ in $G[S]$:*

(a) For a forget bag $X_i = X_j \setminus \{v\}$ with child X_j, $\alpha - v$ is the CCC of $\Sigma \setminus \{v\}$ in $G[S]$, and $\Sigma \setminus \{v\} \subseteq S \subseteq V_i \setminus (X_i \setminus \Sigma)$.

(b) For an introduce bag $X_i = X_j \cup \{v\}$ with child X_j, $\alpha + v$ is the CCC of $\hat{\Sigma} = \Sigma \cup \{v\}$ in $\hat{S} = S \cup \{v\}$, and $\hat{\Sigma} \subseteq \hat{S} \subseteq V_i \setminus (X_i \setminus \hat{\Sigma})$.

Lemma 2 (Lemma 10 in [2])**.** *Let X_i be a join bag of T with children $X_{i'}$ and $X_{i''}$ and let $\Sigma \subseteq X_i = X_{i'} = X_{i''}$. Suppose there exist S' and S'' such that $\Sigma \subseteq S' \subseteq V_{i'}$, $\Sigma \subseteq S'' \subseteq V_{i''}$. If α', α'' are CCCs of Σ in $G[S']$, $G[S'']$, then $\alpha = \alpha' + \alpha''$ is the CCC of Σ in $G[S' \cup S'']$.*

3.3 Dynamic programming

We consider a nice tree decomposition of G with bags $\{X_i\}$ and perform a recursion from the leaves to the root node of the tree decomposition by properly combining the partial solutions obtained in each bag of the tree. Given a bag $X_i \in T$, let Σ be the intersection of a partial critical set $S \subseteq V_i$ with X_i. The corresponding complementary sets are $\bar{\Sigma} = X_i \setminus \Sigma$ and $\bar{S} = V_i \setminus S$. Therefore, $\bar{S}$ represents

the remaining nodes in V_i after the removal of the critical set S, and the pairwise connectivity of $G[V_i]$ is thus given by the residual graph $G[\bar{S}]$. We consider in the DP algorithm CCCs that refer to both (Σ, S) (as described above with a CCC α) and $\bar{\Sigma}, \bar{S}$. In particular, for a potential CCC $\beta \in \Gamma(\bar{\Sigma}, |V_i \setminus X_i|)$, we consider a realisation $G[\bar{S}]$ of β, i.e. such that

$$\beta = \{(\bar{\Sigma} \cap V(H), |V(H) \setminus \bar{\Sigma}|) \colon H \in \mathrm{comp}(G[\bar{S}]), \bar{\Sigma} \cap V(H) \neq \emptyset\}.$$

Similarly to the previous notations for α, we adopt the general notation $\beta = \{(B_i, b_i)\}_{i=1\ldots k}$. The following definition will represent the number of connected pairs (i.e., the pairwise connectivity) of the components in $G[\bar{S}]$ that have at least one node in $\bar{\Sigma}$:

$$\|\beta\| \stackrel{\text{def}}{=} \sum_{(B_i,b_i)\in\beta} \binom{|B_i| + b_i}{2}.$$

For each bag X_i, we define $f_i(\Sigma, \alpha, \beta, \pi, p)$ as the minimum sum of the deletion costs of nodes in S, denoted as $k(S)$, where S is such that

(i) $S \cup \bar{S}$ is a partition of V_i, $\Sigma \subseteq S \subseteq V_i \setminus \bar{\Sigma}$, $\bar{\Sigma} \subseteq \bar{S} \subseteq V_i \setminus \Sigma$,

(ii) $G[S]$ is a realisation of α, $|\mathrm{comp}(G[S])| = p$,

(iii) $G[\bar{S}]$ is a realisation of β, $\#\mathrm{conn}(G[\bar{S}]) = \pi$.

Let $\Pi_i(\Sigma, \alpha, \beta, \pi, p)$ be the above optimisation problem.

Optimal value. The optimal value is recovered by reading the states $f_1(\Sigma, \alpha, \beta, \pi, p)$ with $f_1 \leq K$, $\alpha = \{(A_1, a_1)\}$ or $\alpha = \emptyset$ with $\pi \leq \binom{|V|}{2}$, $p = 1$ and selecting the state with π as small as possible. Note that requiring $p = 1$ forces the deleted set to be connected since there is only one connected component inside S. We also remark that an optimal solution can be easily recovered from the DP entries by implementing a backtracking procedure.

We iteratively compute the f_i values by going from the leaves up to the root bag X_1 through the following recursions. For the ease of exposition, we provide the proof of correctness of the recursions in Appendix A.

Start bag. A start bag is a singleton $X_i = \{v\}$. We have

$$f_i(\Sigma, \alpha, \beta, \pi, p) = \begin{cases} k_v & \text{if } \Sigma = \{v\},\ \alpha = \{(\{v\}, 0)\},\ \beta = \emptyset,\ \pi = 0,\ p = 1 \\ 0 & \text{if } \Sigma = \emptyset,\ \alpha = \emptyset,\ \beta = \{(\{v\}, 0)\},\ \pi = 0,\ p = 0 \\ +\infty & \text{in all other cases.} \end{cases}$$

Forget bag. Bag X_i has one child bag X_j, with $X_i = X_j \setminus \{v\}$. We compute $f_i(\Sigma, \alpha, \beta, \pi, p)$ as follows.

$$f_i(\Sigma, \alpha, \beta, \pi, p) = \min\Big\{\{f_j(\Sigma \cup \{v\}, \alpha', \beta, \pi, p) \colon \alpha' - v = \alpha\} \cup \{f_j(\Sigma, \alpha, \beta', \pi, p) \colon \beta' - v = \beta\}\Big\} \quad (5)$$

Introduce bag. Bag X_i has one child bag X_j, with $X_i = X_j \cup \{v\}$. In this case, we compute $f_i(\Sigma, \alpha, \beta, \pi, p)$ as follows.

$$f_i(\Sigma, \alpha, \beta, \pi, p) = \begin{cases} \min\{f_j(\Sigma \setminus \{v\}, \alpha', \beta, \pi, p') \colon \\ \qquad \alpha' + v = \alpha,\ p = p' + |\alpha| - |\alpha'|\} + k_v & (v \in \Sigma) \\ \min\{f_j(\Sigma, \alpha, \beta', \pi', p) \colon \\ \qquad \beta' + v = \beta,\ \pi = \pi' + \|\beta\| - \|\beta'\|\} & (v \notin \Sigma) \end{cases} \quad (6)$$

Join bag. Bag X_i has two child bags $X_{i'}$, $X_{i''}$ and $X_i = X_{i'} = X_{i''}$. The recursion for $f_i(\Sigma, \alpha, \beta, \pi, p)$ is written as follows.

$$\begin{aligned} f_i(\Sigma, \alpha, \beta, \pi, p) = \min \Big\{ & f_{i'}(\Sigma, \alpha', \beta', \pi', p') + f_{i''}(\Sigma, \alpha'', \beta'', \pi'', p'') - k(\Sigma): \\ & \alpha' + \alpha'' = \alpha,\ \beta' + \beta'' = \beta, \\ & p' + p'' - |\alpha'| - |\alpha''| + |\alpha| = p, \\ & \pi' + \pi'' - \left\|\beta'\right\| - \left\|\beta''\right\| + \|\beta\| = \pi \Big\} \end{aligned} \tag{7}$$

Proposition 3: The Connected Pairwise CNP with unit connection costs is solved to optimality by the recursive functions (5)-(7) over any nice tree decomposition of a given graph.

Proof. See Appendix A. □

Theorem 1: The Connected Pairwise CNP with unit connection costs can be solved in polynomial time over graphs with a treewidth bounded by a constant κ.

Proof. If the treewidth of a graph is bounded by κ, we can find a nice tree decomposition of the graph with a number of bags $\mathcal{O}(\kappa n)$ in polynomial time.

The DP algorithm requires to compute the values $f_i(\Sigma, \alpha, \beta, \pi, p)$ for all bags X_i, all $\Sigma \subseteq X_i$, all $\alpha \in \Gamma(\Sigma, |V_i \setminus X_i|)$, $\beta \in \Gamma(\bar{\Sigma}, |V_i \setminus X_i|)$, all values of π and p with $0 \leq \pi \leq |V_i|(|V_i|-1)/2$ and $0 \leq p \leq |V_i|$. As $|V_i| \leq n$, only $n(n-1)/2 + 1$ values are possible for parameters π and only $n+1$ values for parameter p. Since κ is a constant, the number of possible bags is in $\mathcal{O}(n)$ and each bag X_i contains a constant number of nodes bounded by $\kappa + 1$. So there are at most $2^{\kappa+1}$ sets Σ and, for any $\Sigma \subseteq X_i$, both $|\Sigma|$ and $|\bar{\Sigma}|$ are $\leq \kappa + 1$. As noticed in [2], the latter conditions imply that the number of potential CCCs α (β) is bounded by a polynomial function in n: the number of the different partitions A_i (B_i) is a function of κ and, for each partition, the possible choices of elements $0 \leq a_i$ (b_i) $\leq n$ is bounded by $(n+1)^{(\kappa+1)}$. Hence, the number of f_i values to be computed is bounded by a polynomial in n.

Finally, since all the potential CCCs can be enumerated in polynomial time, it is easy to observe that computing each entry of the recursive functions (5)-(7) requires only a polynomial number of operations. Therefore, the DP algorithm has a polynomial time complexity when κ is a constant. □

We close this section by underlining that in the case of arbitrary connection costs, the value of the parameter π would not be polynomially bounded anymore. Moreover, the computation of the connection cost of a connected component is less straightforward and would invalidate the recursion relations (6) and (7).

4 The Connected CNP over trees

This section investigates the complexity for the Connected CNP over trees. Since trees have a bounded treewidth (equal to 1), the results of the previous section apply to the instances of the Connected CNP over trees with unit connection costs and to the instances of the Connected *MinMaxC*/*MaxNum* CNP over trees. However, while our DP algorithm over a nice tree decomposition of a graph has a higher

time complexity than the algorithm of [2] for the classic CNP, we will show in the following that exploiting the connected structure of the critical set leads to faster specialised algorithms than those for the classic CNP over trees.

In this section we first prove that the Connected Pairwise CNP over trees with arbitrary connection and deletion costs is NP-complete. Then, we present pseudopolynomial and polynomial algorithms for different input configurations and compare the obtained results with the existing results for the Pairwise CNP over trees in [18]. Our results are summarised in Table 1 together with the results of [18] on the classic CNP over trees for comparison. We also report in the table the results for the problems over paths. In Appendix B, we also prove that the Connected *MinMaxC* CNP over trees turns out to be polynomially solvable. As the Connected *MaxNum* CNP over trees can be solved in polynomial time by the DP algorithm developed for graphs with bounded treewidth without imposing any restriction on the input data (see Appendix C), we decided not to analyse further this problem variant over trees.

c_{ij}	k_i	Complexity	
		Connected Pairwise CNP	Pairwise CNP
Trees:			
≥ 0	> 0	Weakly NP-hard	Strongly NP-hard [18]
≥ 0	$= 1$	Solvable in $\mathcal{O}(n^3)$	Strongly NP-hard [18]
$= 1$	> 0	Solvable in $\mathcal{O}(n^4)$	Solvable in $\mathcal{O}(n^7)$ [18]
Paths:			
≥ 0	> 0	Solvable in $\mathcal{O}(n^3)$	Weakly NP-hard [7, 18]

Table 1: Summary of the derived complexity results for the Connected Pairwise CNP over trees/paths and comparison with the complexity results in the literature for the Pairwise CNP over trees/paths.

4.1 NP-completeness

In [18] it has been shown that the Pairwise CNP over trees with general connection costs ($c_{ij} \geq 0$) is strongly NP-hard. It is also shown that the same result holds even with the further input restriction $k_i = 1$ for each $i \in V$. The Pairwise CNP over paths with general costs has been proved to be NP-complete in [18] and shown to be weakly NP-complete in [7]. We show here that the decision version of the Connected Pairwise CNP over trees, where we seek for solutions with a pairwise connectivity less than (or equal to) a threshold value Π, is also NP-complete.

Theorem 2: The decision version of the Connected Pairwise CNP over trees with arbitrary connection and deletion costs is NP-complete.

Proof. The decision version of the Connected Pairwise CNP over trees is indeed an NP-problem as we can compute and check the objective value of any solution in polynomial time. We prove the theorem by reducing an arbitrary instance of the decision version of the Knapsack Problem (KP), a well known NP-complete problem, to an instance of the decision version of the Connected Pairwise CNP over trees. In an instance of the decision version of the KP we have n items, each with a profit p_i and a weight w_i for $i = 1, \ldots, n$, and a threshold value P. The problem asks whether there exists a subset of the items with a total weight that does not exceed a capacity W and a total profit greater than (or

equal to) P. If we consider binary variables x_i equal to 1 if item i is chosen and equal to 0 otherwise, we need to find a solution vector x^* such that $\sum_{i=1}^{n} w_i x_i^* \leq W$ and $\sum_{i=1}^{n} p_i x_i^* \geq P$.

We now construct an instance of the Connected Pairwise CNP as follows. For each item i, we introduce two nodes $2i-1$ and $2i$ linked by an edge $\{2i-1, 2i\}$. Each node $2i-1$ is a leaf node with $k_{2i-1} = 1$. We set $k_{2i} = w_i$ for each node $2i$ and $c_{2i-1,2i} = p_i$ for each pair $(2i-1, 2i)$. We then add a root node $2n+1$ with $k_{2n+1} = 1$ and an edge $\{2i, 2n+1\}$ to each node $2i$ with $c_{2i,2n+1} = \Pi + 1$. All other connection costs are set to 0. Finally, we set $K = W + 1$ and $\Pi = \sum_{i=1}^{n} p_i - P$. Obviously, such a reduction is polynomial.

We show that there exists a feasible solution for the decision version of the KP if and only if there exists a feasible solution for the decision version of the Connected Pairwise CNP. Let v^* denote a feasible solution vector of the Connected Pairwise CNP with binary variables v_j ($j = 1, \ldots, 2n+1$) equal to 1 if node j is removed. Given a solution x^* to the decision version of the KP, we derive the corresponding Connected Pairwise CNP solution by setting $v_{2i}^* = x_i^*$, $v_{2i-1}^* = 0$ for $i = 1, \ldots, n$ and $v_{2n+1}^* = 1$. Such a solution is feasible because it has a connected set of removed nodes, $\sum_{j=1}^{2n+1} k_j v_j^* \leq W + 1 = K$ and a pairwise connectivity equal to

$$\sum_{i=1}^{n} p_i - \sum_{i:v_{2i}^*=1} p_i \leq \sum_{i=1}^{n} p_i - P = \Pi.$$

On the other hand, suppose we have a solution v^* to the decision version of the Connected CNP. This solution must necessarily have $v_{2n+1}^* = 1$ or else the objective function would be larger than Π (as in that case we could remove at most one intermediate node $2i$ in order to maintain the connectivity of the critical set). We can also assume without loss of generality that $v_{2i-1}^* = 0$ for $i = 1, \ldots, n$. In fact, as $v_{2n+1}^* = 1$, any leaf node $2i-1$ can be removed only if its neighbour $2i$ is removed. But in this case the removal of a leaf node is suboptimal as this operation would only induce budget consumption without lowering the objective function. We derive a feasible KP solution x^* by setting $x_i^* = v_{2i}^*$ for $i = 1, \ldots, n$. Indeed, we have:

$$\sum_{i=1}^{n} w_i x_i^* = \sum_{i=1}^{n} w_i v_{2i}^* \leq W \qquad \text{and} \qquad \sum_{i=1}^{n} p_i x_i^* = \sum_{i:v_{2i}^*=1} p_i \geq P$$

where the inequality $\sum_{i:v_{2i}^*=1} p_i \geq P$ is implied by considering that for the objective function of v^* we have $\sum_{i=1}^{n} p_i - \sum_{i:v_{2i}^*=1} p_i \leq \Pi = \sum_{i=1}^{n} p_i - P$.

Hence, since the decision version of the KP can be reduced to the decision version of the Connected Pairwise CNP over trees, the Connected Pairwise CNP over trees is NP-complete. □

4.2 Dynamic Programming algorithms

We derive dynamic programming algorithms to solve the Connected Pairwise CNP over trees. The logic of the algorithms is to recursively exploit the requirement that the set of the removed nodes must be connected, which implies that over trees the removed nodes must constitute a tree as well. This consideration leads to dynamic programming schemes running with a considerably lower time complexity with respect to the solution methods derived in [18] for the Pairwise CNP over trees. The proposed dynamic programs allow us to show that the Pairwise CNP over trees with arbitrary connection and deletion costs is weakly NP-hard and that the problem is polynomially solvable when either the connection costs or the deletion costs are one. Hence, we make progress in evaluating at

what extent the inherent difficulty of solving the Pairwise CNP over trees reduces when the set of the deleted nodes is required to be connected.

Consider a tree $\mathcal{T}$ rooted in a given node r. We define the levels of $\mathcal{T}$ as follows. The children of the root node r constitute the first level of $\mathcal{T}$, the children of the children of r define the second level of $\mathcal{T}$, and we proceed in the same way moving down in $\mathcal{T}$ until all its levels are identified. For our algorithmic developments, we denote by $\mathcal{T}_a$ the subtree of tree $\mathcal{T}$ rooted in a node a. Any subtree $\mathcal{T}_a$ has node a as root, it contains the children of a and all the subsequent nodes in the lower levels of $\mathcal{T}$. We denote the $i-th$ child of a as a_i and the set of children of a as CH_a, i.e. $CH_a = \{a_1, a_2, \dots\}$. Also, let $n_a = |\mathcal{T}_a|$ be the number of nodes in $\mathcal{T}_a$. We indicate by $PW(\mathcal{T}_a) = \sum_{i<j}\{c_{ij} : i, j \in \mathcal{T}_a\}$ the overall cost of the connections between each node pair in $\mathcal{T}_a$. Notice that the computation of all $PW(\mathcal{T}_a)$ can be performed in $\mathcal{O}(n^3)$. Finally, let f_i denote the father node of a node i.

The structure of the dynamic programming recursions will be based on a suitable exploration of each subtree $\mathcal{T}_a$ where node a is removed and any other node can be removed only if all the nodes in the unique path between the root a and it are removed. We remark that our dynamic programming schemes resemble the so called *left-right approach* proposed in [29] and a related variant proposed in [16] for the precedence constraint knapsack problem over trees. This problem is a variant of the standard knapsack problem where the items are associated with the nodes of a tree and an item can be packed only if all its predecessors in the unique path from the root node are selected. In the following, we first handle the case with arbitrary connection costs c_{ij} and then consider the case with unit connection costs.

4.2.1 Arbitrary connection costs ($c_{ij} \geq 0$ for $i, j \in V$)

For each subtree $\mathcal{T}_a$, we compute an optimal solution that minimises the pairwise connectivity in the subtree when root a is removed. We first index the nodes by a depth first search starting from the root, i.e. $a = 1$, and excluding the leaves. Let n'_a ($< n_a$) denote the overall number of ordered nodes. Then, we recursively consider the nodes for removal according to the depth first search (see the example in Figure 1 after the description of the dynamic program functions). The leaves in $\mathcal{T}_a$ will not be visited as if their father node is removed in a solution, the removal of any leaf node will not contribute to further lowering the objective function. If instead the father node is not removed, no leaf node can be removed as the resulting set of deleted nodes would not be connected. For each subtree $\mathcal{T}_a$, we consider the subproblems where a subset of the nodes can be removed. Correspondingly, we define for a triple (i, j, k) with $1 \leq i \leq j \leq n'_a$ and $0 \leq k \leq K$:

$R_a(i, j, k)$:= minimum connection costs of a solution for subtree $\mathcal{T}_a$ when the nodes in the *path* from 1 to i are removed and other nodes in set $\{1, \dots, j\}$ can be removed, given a budget k.

If no feasible solution exists for a triple (i, j, k), we set $R(i, j, k) = +\infty$ (i.e. an arbitrarily large value). We analyse all the subtrees and compute an optimal solution for the original tree $\mathcal{T}$ by selecting the node $a^* \in V$ such that:

$$a^* = \arg\min\{R_a(1, n'_a, K) + PW(\mathcal{T} \setminus \mathcal{T}_a) : a \in V\} \tag{8}$$

where the connection costs $R_a(1, n_a, K)$ in $\mathcal{T}_a$ are summed with the connection costs $PW(\mathcal{T} \setminus \mathcal{T}_a)$ in the remaining part of the tree. Notice that when a is a leaf node or a node whose children are leaves, we have $n'_a = 1$ and either $R_a(1, n'_a, K) = 0$ if $k_a \leq K$ or else $R_a(1, n'_a, K) = +\infty$. In the remaining cases, to compute $R_a(1, n'_a, K)$ we can recursively compute the $R_a(i, j, k)$ values for some triples (i, j, k) identified by the depth first exploration of $\mathcal{T}_a$. In particular, we will only consider

specific values of j for given values of i and k. For the root node $a = 1$ and $k = 0, \ldots, K$ we have:

$$R_a(1,1,k) = \begin{cases} \sum_{h \in CH_1} PW(\mathcal{T}_h) & \text{if } k_1 \leq k; \\ +\infty & \text{otherwise.} \end{cases} \tag{9}$$

In fact, if the value k of the budget allows the removal of node 1, the connection costs are equal to the sums of the pairwise connectivities of the subtrees rooted in each child of node 1, i.e. $\sum_{h \in CH_1} PW(\mathcal{T}_h)$. Then, we consider the removal of a node $i > 1$ by going down in the tree and applying a *down move*. More precisely, in a *down move* we set $j = i$ and have for $k = 0, \ldots, K$:

$$R_a(i,j,k) = \begin{cases} R_a(f_i, j-1, k-k_i) - PW(\mathcal{T}_i) + \sum_{h \in CH_i} PW(\mathcal{T}_h) & \text{if } k_i \leq k; \\ +\infty & \text{otherwise.} \end{cases} \tag{10}$$

Hence, in a *down move* the node i is added to the set of nodes available for removal ($j = i$) and node i is removed when $k_i \leq k$. In such a case, the pairwise connectivity in $\mathcal{T}_a$ is given by the connection costs obtained with triple $(f_i, j-1, k-k_i)$, without the term $PW(\mathcal{T}_i)$, plus the connection costs in the subtrees rooted in each child of i. The latter costs are represented by the term $\sum_{h \in CH_i} PW(\mathcal{T}_h)$.

A *down move* is performed whenever we can visit a new node by moving down in $\mathcal{T}_a$. Otherwise, after visiting a node i, we backtrack to its father node f_i and apply an *up move* where we have for $k = 0, \ldots, K$:

$$R_a(f_i, j, k) = \min\{R_a(f_i, i-1, k);\ R_a(i, j, k)\}. \tag{11}$$

In the *min* comparison in (11), we evaluate, for a given value of k and father node f_i, whether it is convenient to remove the child node i when the nodes available for removal are the nodes $1, \ldots, j$. If node i is removed, we consider the term $R_a(i, j, k)$. Else, we consider the subproblem associated with triple $(f_i, i-1, k)$ and take the term $R_a(f_i, i-1, k)$ as the nodes $i+1, \ldots, j$ (which lie in the tree under the node i when an *up move* is considered) cannot be available for removal if node i is not removed. An *up move* is performed until a new *down move* can be applied. By following these rules, the tree is explored until we compute the optimal solution value $R_a(1, n'_a, K)$ (see the example in Figure 1). Without going into the details, we remark that we can implement a simple backtracking procedure to compute also the optimal solution corresponding to $R(1, n'_a, K)$ without increasing the time complexity of the algorithm. Considering the proposed dynamic program, we can state the following proposition.

Proposition 4: The Connected Pairwise CNP over trees with arbitrary connection and deletion costs admits a pseudopolynomial time algorithm with time complexity $\mathcal{O}(n^2 K)$.

Proof. For a given subtree $\mathcal{T}_a$, the dynamic programming algorithm analyses the relevant edges twice for each value of k. Thus, the number of elementary operations needed to execute the algorithm is bounded by $\mathcal{O}(n_a K)$. Since we consider all the subtrees to compute an optimal solution for $\mathcal{T}$, the overall running time is bounded by $\mathcal{O}(n^2 K)$. □

With the results of Proposition 4 and Theorem 2, we immediately have the following statement.

Corollary 1: The Connected Pairwise CNP over trees with arbitrary connection and deletion costs is weakly NP-hard.

In problem instances with unit deletion costs ($k_i = 1$ for each $i \in V$), we have the following result.

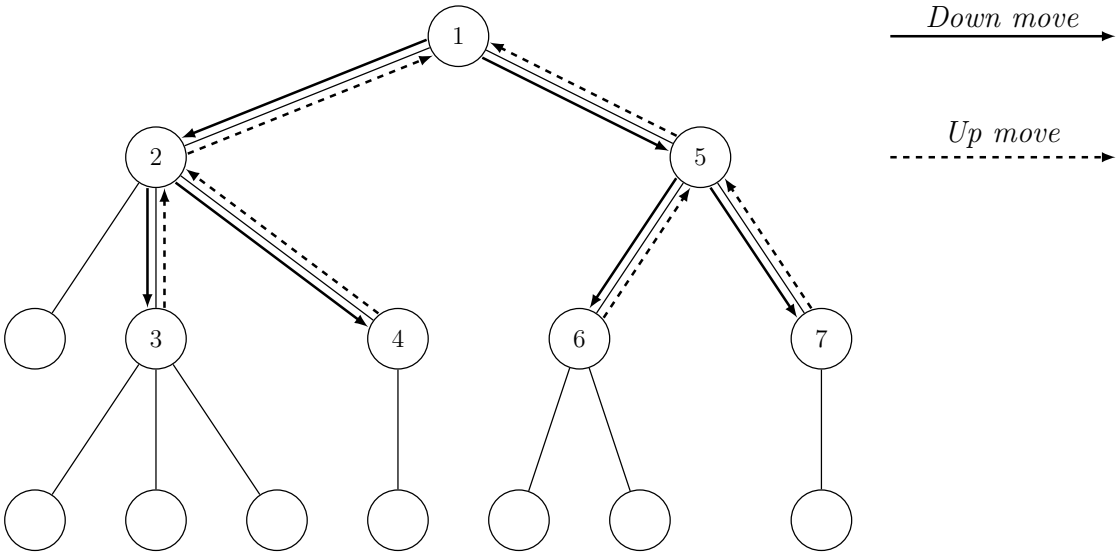

Figure 1: Example of a sequence of *down/up moves* for the exploration of a subtree $\mathcal{T}_a$ where the following sequence of $R_a(i,j,k)$ entries is computed for $k = 0, \ldots, K$: $R_a(1,1,k)$, $R_a(2,2,k)$, $R_a(3,3,k)$, $R_a(2,3,k)$, $R_a(4,4,k)$, $R_a(2,4,k)$, $R_a(1,4,k)$, $R_a(5,5,k)$, $R_a(6,6,k)$, $R_a(5,6,k)$, $R_a(7,7,k)$, $R_a(5,7,k)$, $R_a(1,7,k)$.

Corollary 2: The Connected Pairwise CNP over trees with arbitrary connection costs and unit deletion costs is polynomially solvable.

Proof. Since we have $K < n$ in the meaningful instances with unit deletion costs, the dynamic program described above constitutes a polynomial time algorithm as it runs with $\mathcal{O}(n^3)$ time complexity. □

4.2.2 Unit connection costs ($c_{ij} = 1$ for $i, j \in V$)

In any instance of the Connected Pairwise CNP over trees with unit connection costs, the pairwise connectivity is bounded from above by the number of node pairs $n(n-1)/2$. We can modify the above recursions for dynamic programming by deletion costs and perform dynamic programming by connection costs for a given subtree $\mathcal{T}_a$. For a triple (i,j,c) with $1 \leq i \leq j \leq n'_a$ and $0 \leq c \leq \sum_{h \in CH_1} PW(\mathcal{T}_h)$, we define:

$R_a(i,j,c)$:= minimum deletion cost of a solution for subtree $\mathcal{T}_a$ with connection costs $\leq c$ when the nodes in the path from 1 to i are removed and other nodes in set $\{1, \ldots, j\}$ can be removed.

As in the previous algorithm, we set $R(i,j,c) = +\infty$ if no feasible solution exists for a triple (i,j,c). In the root node $a = 1$ we have for each value of c:

$$R_a(1,1,c) = \begin{cases} k_1 & \text{if } \sum_{h \in CH_1} PW(\mathcal{T}_h) \leq c; \\ +\infty & \text{otherwise.} \end{cases} \tag{12}$$

Clearly, when we only remove the root node, the value of R coincides with k_1 when $\sum_{h \in CH_1} PW(\mathcal{T}_h) \leq$

c. In a *down move* we again set $j = i$ and have for each value of c:

$$R_a(i,j,c) = \begin{cases} R_a(f_i, j-1, c+c') + k_i & \text{if } R_a(f_i, j-1, c+c') + k_i \leq K, \\ & \text{with } c' = PW(\mathcal{T}_i) - \sum_{h \in CH_i} PW(\mathcal{T}_h) \text{ and } c + c' \leq \sum_{h \in CH_1} PW(\mathcal{T}_h); \\ +\infty & \text{otherwise.} \end{cases} \tag{13}$$

Here, we consider the removal of node i in a feasible solution for triple (i,j,c). Correspondingly, we add the deletion cost k_i to the deletion cost given by the triple $(f_i, j-1, c+c')$ (with $j = i$), where c' represents the decrease of the pairwise connectivity when node i is removed. Clearly, a value $c + c'$ out of the relevant range of values of c is not considered. Similarly to the previous dynamic program, in an *up move* we have for each c:

$$R_a(f_i, j, c) = \min\{R_a(f_i, i-1, c);\ R_a(i, j, c)\}. \tag{14}$$

The optimal value for the subtree $\mathcal{T}_a$ is given by the smallest quantity c_{min} such that $R_a(1, n'_a, c_{min}) \leq K$. The corresponding solution value for $\mathcal{T}$ is $c_{min} + PW(\mathcal{T} \setminus \mathcal{T}_a)$. By analyzing all the subtrees, we identify the subtree $\mathcal{T}_a$ that contains a global optimal solution and recover such a solution by a backtracking procedure. Considering the modified dynamic program, we state the following proposition.

Proposition 5: The Connected Pairwise CNP over trees with unit connection costs admits a polynomial time algorithm with time complexity $\mathcal{O}(n^4)$.

Proof. The dynamic programming algorithm for each subtree $\mathcal{T}_a$ has a running time of $\mathcal{O}(2n_a \cdot n_a(n_a - 1)/2) = \mathcal{O}(n_a^3)$ as we consider the relevant edges in the subtree twice for each value of c and we have $c \leq n_a(n_a - 1)/2$. Considering all the $\mathcal{O}(n)$ subtrees, the overall time complexity is $\mathcal{O}(n^4)$. □

4.2.3 Polynomiality over paths

The Pairwise CNP was suggested to be weakly NP-hard over paths with arbitrary connection and deletion costs in [18]. This result was then proved in [7]. We show here that the Connected Pairwise CNP over paths can be easily solved in polynomial time. We consider the nodes indexed by increasing numbers from left to right in the path. It suffices to evaluate n solutions where we start removing a different node $i \in V$ and progressively delete subsequent nodes $j > i$ until the deletion budget allows it. Since the connection costs of a given solution can be computed in $\mathcal{O}(n^2)$, the problem can be solved to optimality in $\mathcal{O}(n^3)$ time. We summarise this result in the following proposition.

Proposition 6: The Connected Pairwise CNP over paths is solvable in polynomial time with time complexity $\mathcal{O}(n^3)$.

Notice that the same algorithm can be applied over paths to the other objective functions studied in this article, *MinMaxC* and *MaxNum*, but with a lower time complexity due to the computation of the objective function, i.e., $\mathcal{O}(n^2)$ for the Connected *MinMaxC* CNP and $\mathcal{O}(n)$ for the Connected *MaxNum* CNP (whose solutions can yield either one or two connected components).

We conclude this section by observing that even stronger results can be obtained for the version of the CNP where the set of critical nodes is further restricted to form a path, see e.g. [25]. In such a

version of the CNP over trees, since there exists a unique path between any two nodes, there is only a polynomial number of $\mathcal{O}(n^2)$ paths that we can consider for the critical set, therefore solving such a problem over trees can be done in polynomial time.

5 The Connected CNP over series-parallel graphs

Several variants of the CNP have been studied specifically over series-parallel graphs: for example, [40] established the polynomiality of the CNP based on the cardinality of the largest connected component or the number of connected components, while [7] established polynomiality for a certain class of the Distance-CNP with unit connection costs. Since series-parallel graphs have treewidth 2, they can be solved by the (pseudo-)polynomial algorithm described in Section 3 in certain cases. In this section, we provide further insight on the cases where such instances prove to be NP-hard.

We recall a few definitions and facts about series-parallel graphs. *Two-terminal graphs* (TTG) $G(s,t)$ with source node s and sink node t represent the building blocks of a series-parallel graph, since such a graph can be obtained by performing *series and parallel compositions* of two TTG at a time: the series operation consists in merging nodes t_1 and s_2 of TTG graphs $G_1(s_1,t_1)$ and $G_2(s_2,t_2)$ while the parallel operation consists in merging node s_1 with s_2 on one part and node t_1 with t_2 on the other part. The full series-parallel graph G can be constructed by a set of series-parallel operations which can be represented by a binary tree where the leaves are two-nodes single-edged graphs called $\hat{K}_2$ and the construction can be performed by following the operations from leaves to root (each node being either a series or a parallel operation between the graphs of the nodes below). Such a binary tree can be identified in linear time [43, 44]. An example is given in Figure 2 where two TTGs are merged using a parallel operation.

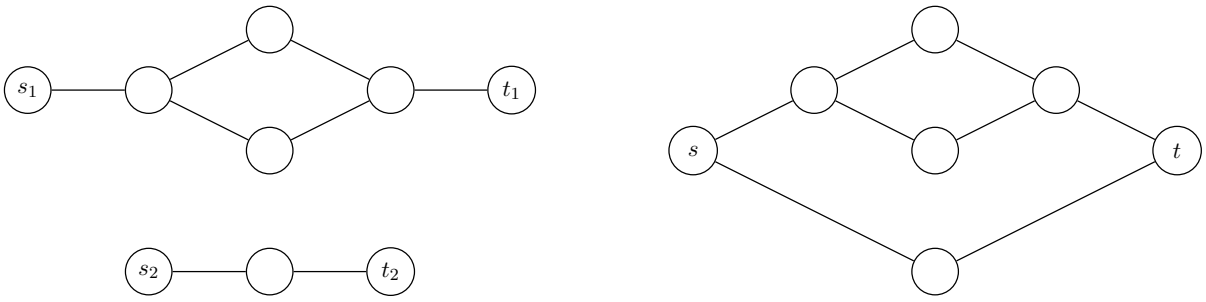

Figure 2: Example of a parallel operation between two TTGs of a series-parallel graph (on the left), where the final result is displayed on the right.

We now propose to study the complexity of the Connected Pairwise CNP over series-parallel graphs when the connection costs are non-unitary, since the case of unitary connection costs is handled by the recursion proposed in Section 3. We will first establish the NP-completeness of the Connected CNP based on pairwise connectivity over series-parallel graphs with integer connection and deletion costs and provide a result of strong NP-hardness for the classic Pairwise CNP over series-parallel graphs with integer connection costs. We sum up our findings over series-parallel graphs in Table 2 along with existing results for the classic Pairwise CNP.

We first perform a reduction of the decision version of the Partition Problem to prove the NP-completeness of the Connected CNP over series-parallel graphs.

c_{ij}	k_i	complexity	
		Connected Pairwise CNP	Pairwise CNP
≥ 0	> 0	NP-hard	Strongly NP-hard
$= 1$	> 0	Polynomial	Polynomial [2]

Table 2: Summary of our complexity results for the Connected Pairwise CNP over series-parallel graphs along with complexity results for the Pairwise CNP (partially taken from [2]).

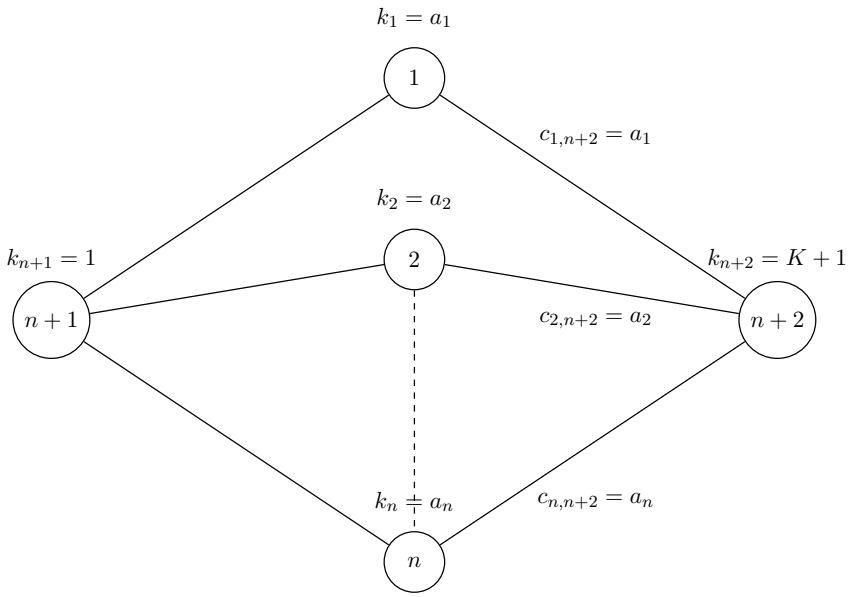

Figure 3: Example of a series-parallel instance for the Connected CNP over series-parallel graphs obtained by reduction of a Partition instance.

Theorem 3: The decision version of the Connected Pairwise CNP over series-parallel graphs is NP-complete with integer connection and deletion costs.

Proof. We introduce an instance of the Partition Problem [23] as follows. An instance of the Partition Problem is made of a set $A = \{1, \ldots, n\}$ of n positive integers a_i and its decision version consists in identifying a subset $A' \subset A$ such that $\sum_{i \in A'} a_i = \sum_{i \in A} a_i/2$.

We construct an instance of the decision version of the Connected CNP based on a general instance of the Partition Problem as follows. For each $i \in A$, we define a node i with $k_i = a_i$. We also define a node $n+1$ with $k_{n+1} = 1$ and a node $n+2$ with $k_{n+2} = \sum_{i \in A} a_i/2 + 2$ and add edges $(i, n+1)$ and $(i, n+2)$ for $i \in A$. We set $c_{i\,n+2} = a_i$ for $i \in A$ while all other connection costs are null. The terminal nodes s and t are nodes $n+1$ and $n+2$ and the overall graph is a series-parallel graph. The maximum threshold K on the total weight of deleted nodes is fixed at $K = \sum_{i \in A} a_i/2 + 1$ and the maximum threshold Π on total pairwise connectivity is fixed at $P = \sum_{i \in A} a_i/2$ for the decision version of the Connected CNP. We display a graphical example of the obtained Connected CNP instance in Figure 3 where it can be seen that the graph is obtained by series combinations of $\hat{K}_2$ graphs and finish the combination by an overall parallel operation. The reduction is polynomial.

We now prove that there exists a feasible solution for the decision version of the Partition Problem iff there exists a feasible solution for the decision version of the Connected CNP. Let us consider a solution A^* to the decision version of the Partition Problem and define the corresponding Connected CNP solution as $v_i^* = i$ for $i \in A^*$, $v_{n+1}^* = 1$ and $v_{n+2}^* = 0$. Such a solution is connected and has:

$$\sum_{i=1}^{n+2} k_i v_i^* = \sum_{i \in A^*} a_i + 1 = K \qquad \text{and} \qquad \sum_{i<j} c_{ij} u_{ij}^* = \sum_{i \in A \setminus A^*} a_i = \sum_{i \in A} a_i/2 = P,$$

which implies it is a feasible solution to the decision version of the Connected CNP. On the other hand, suppose that we have a feasible solution to the decision version of the Connected CNP. It has necessarily $v_{n+2}^* = 0$ (in order to respect the budget constraint). Since the critical set must be connected, setting $v_{n+1}^* = 0$ would mean that we can delete only one node in the graph. This could result in an optimal solution only if, given the index $i_m \in A$, we have $a_{i_m} = \sum_{i \in A} a_i/2$. In such a case, the deleted node corresponds to the optimal subset A^* of the Partition Problem and the correspondence is easy to obtain. In any other case, any feasible solution to the decision version of the Connected CNP must have $v_{n+1}^* = 1$ so that the deletion of any subset of the nodes $i \leq n$ induces a connected subgraph. The residual budget for deleting such nodes is $K - 1 = \sum_{i \in A} a_i/2$. By construction of the connection cost, we have that $\sum_{i,j \in V: i<j} c_{ij} u_{ij}^* = \sum_{i \in A: v_i^* = 0} a_i = \sum_{i \in A} a_i - \sum_{i \in A: v_i^* = 1} a_i$, hence we have that $\sum_{i,j \in V: i<j} c_{ij} u_{ij}^* \geq \sum_{i \in A} a_i - K + 1 = \sum_{i \in A} a_i/2$. In order to have a Yes instance for the Connected CNP, we need to have the relation:

$$\sum_{i \in A: v_i^* = 1} a_i = \sum_{i \in A: v_i^* = 0} a_i = \sum_{i \in A} a_i/2,$$

which guarantees that the partition instance is a Yes instance too, which completes the proof. □

Since one of our goals is to compare the complexity of the Connected CNP with the complexity of the classic CNP, we will also derive an NP-completeness result for the classic Pairwise CNP over series-parallel graphs, albeit stronger than the one above. In order to do this, we will use the reduction from the Multicut in Trees (MCT) presented in [18] to derive the strong NP-completeness of the Pairwise CNP over trees and we refer to [18] for more details. The MCT consists in finding the edge set of minimum weight whose deletion will disconnect a collection $H = \{\{u_1, v_1\}, \ldots, \{u_t, v_t\}\}$ of node pairs in the tree. In the proof of the following theorem, we consider only the unweighted version of the MCT where each edge has unitary weight, which has been shown to be strongly NP-complete in [24].

Theorem 4: The decision version of the Pairwise CNP is strongly NP-complete on series-parallel graphs with integer connection costs, even with unit deletion costs.

Proof. We need to prove the equivalence of a Yes instance of the MCT with a Yes instance of the Pairwise CNP over series-parallel graphs with unit deletion costs. Consider an instance of the unweighted MCT, i.e. a tree $T = (V, E)$ with $n = |V|$ and a collection H of pairs of nodes to disconnect. An instance of the decision version of the MCT also contains a maximum threshold B and asks whether there exists an edge subset $W \subset E$ with $|W| \leq B$ such that the removal of edge subset W disconnects each pair in H. In order to construct an instance of the decision version of the Pairwise CNP over series-parallel graphs, we will first follow the reduction provided in Section 2 of [18]. Therefore we first construct a tree $T' = (V', E')$ containing all the nodes in V. For each edge of $e = \{i, j\} \in E$, we add two nodes e_1 and e_2 and three edges $\{i, e_1\}$, $\{e_1, e_2\}$ and $\{e_2, j\}$, i.e. $i - e_1 - e_2 - j$ forms a subpath. We set the connection cost $c_{uv} = 1$ for $\{u, v\} \in H$, $c_{uv} = M > |H|$ if $\{u, v\} = \{e_1, e_2\}$ for

all $e \in E$ and $c_{uv} = 0$ in any other case. Finally, we add an extra node q with an edge between q and each leaf node in T' to obtain a series-parallel graph $G(r, q)$ with terminal nodes r and q where r is the root node of T. We set the connection costs $c_{iq} = 0$ for all nodes $i \in V'$ and all deletion costs $k_i = 1$ for $i \in V' \cup \{q\}$. We set the maximum deletion threshold $K = B$ and the maximum connectivity threshold $P = M(n - K - 1)$. We observe that the deletion of terminal node q does not provide any reduction of the total connection in the decision version of the Connected CNP. Therefore, the logic of the proof follows the steps of Proposition 2.1 in [18] and we refer the reader to it for further details. Since the unweighted MCT is strongly NP-complete, it follows that the Connected CNP on series-parallel graphs is strongly NP-complete, even with unit deletion costs. □

The results for the *MinMaxC* CNP over series-parallel graphs are reported in Appendix B.

6 Conclusions

We have introduced in this work a variant of the Critical Node Problem where the set of critical nodes forms a connected subgraph. This requirement had only been partially investigated for specially structured critical sets in [25, 49]. We studied three versions of the problem based on three different objective functions and stated their NP-hardness status on even biconnected planar bipartite graphs. We then studied the complexity of the Connected CNP on tree decompositions of a general graph and derived dynamic programming algorithms. We showed that the problems are polynomially solvable with unit weights in the objective function when the treewidth of the graph is bounded by a constant. We also investigated the considered problems over trees and highlighted the differences in complexity with the classic CNP over trees. Finally, we studied the complexity of the problems over series-parallel graphs with arbitrary costs and we provided additional results on the NP-hardness of several variants of the classic CNP over series-parallel graphs.

The complexity of some versions of the CNP in other types of graphs has been successfully investigated, e.g., in [32] for proper interval graphs and in [31] for bipartite permutation graphs. Therefore, an analysis of the Connected CNP over such specially structured graphs could be a natural direction for future research. Another interesting research line would be the development of decomposition methods based on mathematical programming to solve the Connected CNP on general graphs, as in, e.g., [25] and [49]. Finally, as underlined in the Introduction, many heuristic approaches were developed for the CNP given the inherent difficulty of solving any variant of the problem over general graphs. It would be interesting to tailor some of these approaches to include the constraint of having a connected critical set.

Appendix A Proof of correctness of the DP algorithm over tree decompositions

We consider the following strategy to show the correctness of the recursive functions (5)-(7). For each type of bag of the nice tree decomposition, we show that $f_i(\Sigma, \alpha, \beta, \pi, p)$, i.e. the left hand side (lhs) of the equations, is *at least* as large as the right hand side (rhs) of the equations under the assumption that $f_i(\Sigma, \alpha, \beta, \pi, p)$ exists finite. We also show that $f_i(\Sigma, \alpha, \beta, \pi, p)$ is *at most* as large as a finite value of the rhs of the equations. This implies that the recursive functions are correct when the f_i values are finite. Notice that when the f_i values are not finite, we just have a propagation of $+\infty$ values and thus the correctness of the recursive functions still holds.

Start bag. The recursion for start bags is straightforward: we evaluate the possible removal of node v and consider the corresponding settings of Σ, α, β, π and p.

Forget bag. For a forget bag X_i, we have $V_i = V_j$. First assume that the lhs of (5) exists and is finite. Then there exists an optimal set S for $\Pi_i(\Sigma, \alpha, \beta, \pi, p)$ satisfying the requirements (i)-(iii). As $v \notin X_i$, v is never included in Σ when X_i is considered. We have either $v \in \bar{S}$ or $v \notin \bar{S}$.

Case $v \in \bar{S}$. This implies $v \notin \Sigma$ for the child bag X_j. Let $\bar{\Sigma}' = \bar{\Sigma} \cup \{v\}$, $\Sigma' = \Sigma$; $\Sigma', \bar{\Sigma}'$ form a partition of X_j, and $\bar{\Sigma}' \subseteq \bar{S} \subseteq V_j \setminus \Sigma'$, $\Sigma' \subseteq S \subseteq V_j \setminus \bar{\Sigma}'$. Among all the potential CCCs $\beta' \in \Gamma(\bar{\Sigma}', |V_j \setminus X_j|)$, let us consider the CCC of $\bar{\Sigma}'$ in $G[\bar{S}]$, namely $\beta' = \{(\bar{\Sigma}' \cap V(\hat{H}), |V(\hat{H}) \setminus \bar{\Sigma}'|) \colon \hat{H} \in \mathrm{comp}(G[\bar{S}]), \bar{\Sigma}' \cap V(\hat{H}) \neq \emptyset\}$. By Lemma 1(a) $\beta' - v$ is the CCC of $\bar{\Sigma}$ in $G[\bar{S}]$, namely $\beta' - v = \beta$. Also, α is still the CCC of Σ in $G[S]$, $\#\mathrm{conn}(G[\bar{S}]) = \pi$ and $|\mathrm{comp}(G[S])| = p$. Thus, S is a feasible solution to problem $\Pi_j(\Sigma, \alpha, \beta', \pi, p)$, $\min\{f_j(\Sigma, \alpha, \beta', \pi, p \colon \beta' - v = \beta\}$ exists, finite, and

$$f_i(\Sigma, \alpha, \beta, \pi, p) = k(S) \geq \min\{f_j(\Sigma, \alpha, \beta', \pi, p) \colon \beta' - v = \beta\}.$$

Case $v \notin \bar{S}$. Let $\bar{\Sigma}' = \bar{\Sigma}$, $\Sigma' = \Sigma \cup \{v\}$; $\Sigma', \bar{\Sigma}'$ form a partition of X_j, with $\bar{\Sigma}' \subseteq \bar{S} \subseteq V_j \setminus \Sigma'$, $\Sigma' \subseteq S \subseteq V_j \setminus \bar{\Sigma}'$. Consider $\alpha' = \{(\Sigma' \cap V(\hat{H}), |V(\bar{H}) \setminus \Sigma'|) \colon V(\hat{H}) \in \mathrm{comp}(G[S])\}$, the CCC of Σ' in $G[S]$: by Lemma 1(a) $\alpha' - v$ is the CCC of Σ in $G[S]$, thus $\alpha' - v = \alpha$. Also, β is still the CCC of $\bar{\Sigma}$ in $G[\bar{S}]$, $\#\mathrm{conn}(G[\bar{S}]) = \pi$, $|\mathrm{comp}(G[S])| = p$. Then S is a feasible solution to $\Pi_j(\Sigma \cup \{v\}, \alpha', \beta, \pi, p)$, $\min\{f_j(\Sigma \cup \{v\}, \alpha', \beta, \pi, p \colon \alpha' - v = \alpha\}$ exists, finite, and

$$f_i(\Sigma, \alpha, \beta, \pi, p) = k(S) \geq \min\{f_j(\Sigma \cup \{v\}, \alpha', \beta, \pi, p) \colon \alpha' - v = \alpha\}.$$

This analysis shows that a finite $f_i(\Sigma, \alpha, \beta, \pi, p)$ is not smaller than the rhs of (5). Now assume the rhs of (5) exists, finite, and is given by either $\min\{f_j(\Sigma, \alpha, \beta', \pi, p \colon \beta' - v = \beta\}$ (Case (1)) or $\min\{f_j(\Sigma \cup \{v\}, \alpha', \beta, \pi, p \colon \alpha' - v = \alpha\}$ (Case (2)).

Case (1). Assume that $\min\{f_j(\Sigma, \alpha, \beta', \pi, p \colon \beta' - v = \beta\} = k(S)$ is finite for some S, and let $\bar{\Sigma}' = \bar{\Sigma} \cup \{v\}$ and $\Sigma' = \Sigma$ as $v \notin \Sigma$. By definition of f_j, we have $\bar{\Sigma}' \subseteq \bar{S} \subseteq V_j \setminus \Sigma'$, $\Sigma' \subseteq S \subseteq V_j \setminus \bar{\Sigma}'$, $\#\mathrm{conn}(G[\bar{S}]) = \pi$, $|\mathrm{comp}(G[S])| = p$. By Lemma 1(a) $\beta' - v = \beta$ is the CCC of $\bar{\Sigma} = \bar{\Sigma}' \setminus \{v\}$ in $G[\bar{S}]$. Also, $\Sigma' = \Sigma$ and α is still the CCC of Σ in $G[S]$; $\#\mathrm{conn}(G[\bar{S}]) = \pi$, $|\mathrm{comp}(G[S])| = p$. Hence S is also a feasible solution to $\Pi_i(\Sigma, \alpha, \beta, \pi, p)$, f_i is finite and

$$f_i(\Sigma, \alpha, \beta, \pi, p) \leq k(S) = \min\{f_j(\Sigma, \alpha, \beta', \pi, p \colon \beta' - v = \beta\}.$$

Case (2). Assume $\min\{f_j(\Sigma \cup \{v\}, \alpha', \beta, \pi, p \colon \alpha' - v = \alpha\} = k(S)$ is finite for some S. Let $\Sigma' = \Sigma \cup \{v\}$, $\bar{\Sigma}' = \bar{\Sigma}$. By definition of f_j we have that β is the CCC of $\bar{\Sigma}'$ in $G[\bar{S}]$, α' is the CCC of Σ' in $G[S]$, $\bar{\Sigma}' \subseteq \bar{S} \subseteq V_j \setminus \Sigma'$, $\Sigma' \subseteq S \subseteq V_j \setminus \bar{\Sigma}$, $\#\mathrm{conn}(G[\bar{S}]) = \pi$, $|\mathrm{comp}(G[S])| = p$. By Lemma 1(a) $\alpha' - v = \alpha$ is the CCC of Σ in $G[S]$; β is still the CCC of $\bar{\Sigma}$ in $G[\bar{S}]$. Also, $\#\mathrm{conn}(G[\bar{S}]) = \pi$ and $|\mathrm{comp}(G[S])| = p$. Hence S is a feasible solution to $\Pi_i(\Sigma, \alpha, \beta, \pi, p)$, f_i is finite and

$$f_i(\Sigma, \alpha, \beta, \pi, p) \leq k(S) = \min\{f_j(\Sigma \cup \{v\}, \alpha', \beta, \pi, p \colon \alpha' - v = \alpha\}.$$

The previous inequalities show that $f_i(\Sigma, \alpha, \beta, \pi, p)$ is also bounded from above by a finite rhs of (5). Putting all the results together proves that equality (5) holds for a forget bag.

Introduce bag. We have $v \in X_i$ and $v \notin X_j$. We distinguish two cases for bag X_i: $v \notin \Sigma$, $v \in \Sigma$.

Case $v \notin \Sigma$. Let $f_i(\Sigma, \alpha, \beta, \pi, p)$ exist, finite, and S be the optimal solution to $\Pi_i(\Sigma, \alpha, \beta, \pi, p)$. Considering the child bag X_j, we define $\bar{S}' = \bar{S} \setminus \{v\}$, $S' = V_j \setminus \bar{S} = S$. By the definition of $f_i(\Sigma, \alpha, \beta, \pi, p)$,

α is also the CCC of $\Sigma' = \Sigma$ in $G[S']$. Define β' as the CCC of $\bar{\Sigma}'$ in $G[\bar{S}']$. By Lemma 1(b) $\beta' + v$ is the CCC of $\bar{\Sigma} = \bar{\Sigma}' \cup \{v\}$ in $G[\bar{S}]$, hence $\beta' + v = \beta$. As v is not added to the critical set when $G[V_i]$ is considered, the variation of the pairwise connectivity of $G[V_i]$ and $G[V_j]$ is given by the difference $\|\beta\| - \|\beta'\|$. This because all the neighbours of v in $G[\bar{S}]$ are included in $\bar{\Sigma}$ due to the properties of an introduce bag (see Section 3.1). Thus, we can compute the variation of the pairwise connectivity by considering only the components in $G[\bar{S}]$ and $G[\bar{S}']$ with a non trivial intersection with $\bar{\Sigma}$ and $\bar{\Sigma}'$, respectively. Given this counting argument, if we also have $\pi = \pi' + \|\beta\| - \|\beta'\|$ then $S(= S')$ is a feasible solution to $\Pi_j(\Sigma, \alpha, \beta', \pi', p)$, the rhs of (6) exists, finite, and

$$f_i(\Sigma, \alpha, \beta, \pi, p) = k(S) = k(S') \geq \min\{f_j(\Sigma, \alpha, \beta', \pi', p) \colon \beta' + v = \beta,\ \pi = \pi' + \|\beta\| - \|\beta'\|\}.$$

Now let the rhs of (6) exist, finite, and consider $S' \subseteq V_j$ such that $\min\{f_j(\Sigma, \alpha, \beta', \pi', p) \colon \beta' + v = \beta,\ \pi = \pi' + \|\beta\| - \|\beta'\|\} = k(S')$. We define $\bar{S} = \bar{S}' \cup \{v\}$ and note here that $S = V_i \setminus \bar{S}' = S'$. We have that α is the CCC of Σ in $G[S]$ and $\beta' + v = \beta$ is the CCC of $\bar{\Sigma} \cup \{v\}$ in $G[\bar{S}]$ by Lemma 1(b). If the equality $\pi = \pi' + \|\beta\| - \|\beta'\|$ also holds, we can state that S is a feasible solution to $\Pi_i(\Sigma, \alpha, \beta, \pi, p)$. Hence, f_i is finite and we have

$$f_i(\Sigma, \alpha, \beta, \pi, p) \leq k(S) = \min\{f_j(\Sigma, \alpha, \beta', \pi', p) \colon \beta' + v = \beta,\ \pi = \pi' + \|\beta\| - \|\beta'\|\}.$$

Clearly, the previous two inequalities show that equality (6) holds when $v \notin \Sigma$.

Case $v \in \Sigma$. Let $f_i(\Sigma, \alpha, \beta, \pi, p)$ exist, finite, and S be the optimal solution to $\Pi_i(\Sigma, \alpha, \beta, \pi, p)$. Note that $v \in \Sigma \implies v \notin \bar{S}$. Define $\bar{S}' = \bar{S}$, $S' = V_j \setminus \bar{S}' = S \setminus \{v\}$. Then $G[\bar{S}'] = G[\bar{S}]$ and β is the CCC of $\bar{\Sigma}$ in $G[\bar{S}]$. Note that $\Sigma' = \Sigma \setminus \{v\}$. Define α' as the CCC of Σ' in $G[S']$. Then by Lemma 1(b) $\alpha' + v = \alpha$ is the CCC of Σ in $G[S]$. If the equality $p = p' + |\alpha| - |\alpha'|$ also holds, by employing a counting argument similar to the previous one we can state that S' is a feasible solution to $\Pi_j(\Sigma', \alpha', \beta, \pi, p')$. So we have a finite rhs in (6) and

$$f_i(\Sigma, \alpha, \beta, \pi, p) = k(S) = k(S') + k_v \geq \min\{f_j(\Sigma \setminus \{v\}, \alpha', \beta, \pi, p') \colon \alpha' + v = \alpha,\ p = p' + |\alpha| - |\alpha'|\} + k_v.$$

Let the rhs of (6) exist, finite, and consider a solution S' minimising the rhs. We define $\bar{S} = \bar{S}'$ and $S = V_i \setminus \bar{S} = S' \cup \{v\}$. Note that $\Sigma' = \Sigma \setminus \{v\}$ and $\bar{\Sigma}' = \bar{\Sigma}$. If α' is the CCC of Σ' in $G[S']$, then by Lemma 1(b) $\alpha' + v = \alpha$ is the CCC of Σ in $G[S]$. Also, β is the CCC of $\bar{\Sigma}$ in $G[\bar{S}]$. It is easy to see that if p' is the number of components in $G[S']$ then $p = p' + |\alpha| - |\alpha'|$ is the number of components in $G[S]$. Thus, S is a feasible solution to $\Pi_i(\Sigma, \alpha, \beta, \pi, p)$, f_i is finite and

$$f_i(\Sigma, \alpha, \beta, \pi, p) \leq k(S) = k(S') + k_v = \min\{f_j(\Sigma \setminus \{v\}, \alpha', \beta, \pi, p') \colon \alpha' + v = \alpha,\ p = p' + |\alpha| - |\alpha'|\} + k_v.$$

The previous inequalities prove that equality (6) also holds when $v \in \Sigma$.

Join bag. First, assume that $f_i(\Sigma, \alpha, \beta, \pi, p) = k(S)$ is finite for some $S \subset V_i \setminus \bar{\Sigma}$. Then $\bar{\Sigma} \subseteq \bar{S} \subseteq V_i \setminus \Sigma$, $\Sigma \subseteq S \subseteq V_i \setminus \bar{\Sigma}$, $\#\text{conn}(G[\bar{S}]) = \pi$, $|\text{comp}(G[S])| = p$.

Let us define $\bar{S}' = \bar{S} \cap V_{i'}$, $S' = V_{i'} \setminus \bar{S}'$, $\bar{S}'' = \bar{S} \cap V_{i''}$, $S'' = V_{i''} \setminus \bar{S}''$. We have $\bar{S} = \bar{S}' \cup \bar{S}''$ and $S = S' \cup S''$. Notice also that $\bar{\Sigma} \subseteq \bar{S} \implies \bar{\Sigma} \subseteq \bar{S}'$ since $\bar{\Sigma} \subseteq X_i = X_{i'}$; also, $\bar{S} \subseteq V_i \setminus \Sigma \implies \bar{S} \cap V_{i'} \subseteq V_{i'} \setminus \Sigma$, hence $\bar{\Sigma} \subseteq \bar{S}' \subseteq V_{i'} \setminus \Sigma$. Similarly, it follows that $\Sigma \subseteq S' \subseteq V_{i'} \setminus \bar{\Sigma}$, $\bar{\Sigma} \subseteq \bar{S}'' \subseteq V_{i''} \setminus \Sigma$, $\Sigma \subseteq S'' \subseteq V_{i''} \setminus \bar{\Sigma}$. Let α' and β' denote the CCCs of Σ and $\bar{\Sigma}$ in $G[S']$ and $G[\bar{S}']$, respectively. Also, let α'' and β'' denote the CCCs of Σ and $\bar{\Sigma}$ in $G[S'']$ and $G[\bar{S}'']$, respectively. By Lemma 2, $\alpha' + \alpha''$ is the CCC of Σ in $G[S]$, and $\beta' + \beta''$ is the CCC of $\bar{\Sigma}$ in $G[\bar{S}]$. Hence $\alpha = \alpha' + \alpha''$ and $\beta = \beta' + \beta''$. Let $\pi' = \#\text{conn}(G[\bar{S}'])$, $\pi'' = \#\text{conn}(G[\bar{S}''])$, $p' = |\text{comp}(G[S'])|$, $p'' = |\text{comp}(G[S''])|$. We show now that if we also have

$$\pi = \#\text{conn}(G[\bar{S}]) = (\pi' - \|\beta'\|) + (\pi'' - \|\beta''\|) + \|\beta\|,$$
$$p = |\text{comp}(G[S])| = (p' - |\alpha'|) + (p'' - |\alpha''|) + |\alpha|,$$

then S' and S'' are indeed feasible solutions to $\Pi_{i'}(\Sigma, \alpha', \beta', \pi', p')$ and $\Pi_{i''}(\Sigma, \alpha'', \beta'', \pi'', p'')$, respectively. Let us consider the first equality. As observed in [2], a component of $G[\bar{S}']$ or $G[\bar{S}'']$ that does not intersect $\bar{\Sigma}$ is also a component of $G[\bar{S}]$ with a trivial intersection with $\bar{\Sigma}$ and vice versa. To compute π, we can sum the pairwise connectivity of the components in $G[\bar{S}]$ that do not intersect $\bar{\Sigma}$ with the pairwise connectivity of the components in $G[\bar{S}]$ that intersect $\bar{\Sigma}$. The first term is given by $(\pi' - \|\beta'\|) + (\pi'' - \|\beta''\|)$ as the number of connected pairs in $G[\bar{S}']$ or $G[\bar{S}'']$ belonging to components that intersect $\bar{\Sigma}$ are subtracted from the total number of pairs π' and π''. The second term is given by $\|\beta\|$ as β is the CCC of $\bar{\Sigma}$ in $G[\bar{S}]$. A similar counting argument applies to $p = (p' - |\alpha'|) + (p'' - |\alpha''|) + |\alpha|$ when set Σ and graphs $G[S]$, $G[S']$ and $G[S'']$ are considered.

Since by construction Σ is included in both S' and S'', we have $k(S) = k(S') + k(S'') - k(\Sigma)$. Thus, when S' and S'' are feasible solutions to $\Pi_{i'}(\Sigma, \alpha', \beta', \pi', p')$ and $\Pi_{i''}(\Sigma, \alpha'', \beta'', \pi'', p'')$, the rhs of (7) is finite and we have

$$\begin{aligned} f_i(\Sigma, \alpha, \beta, \pi, p) = k(S) = k(S') + k(S'') - k(\Sigma) \geq \\ \geq \min\{f_{i'}(\Sigma, \alpha', \beta', \pi', p') + f_{i''}(\Sigma, \alpha'', \beta'', \pi'', p'') - k(\Sigma)\}. \end{aligned}$$

Now assume that the rhs of (7) exists, finite, i.e. $f_{i'}(\Sigma, \alpha', \beta', \pi', p') = k(S')$ and $f_{i''}(\Sigma, \alpha'', \beta'', \pi'', p'') = k(S'')$ for a suitable choice of S' and S'' that minimises the rhs. Define $S = S' \cup S''$, $\bar{S} = \bar{S}' \cup \bar{S}''$. Note that here $\Sigma \subset S \subseteq V_i \setminus \bar{\Sigma}$, $\bar{\Sigma} \subseteq \bar{S} \subseteq V_i \setminus \Sigma$. By Lemma 2, since α',α'' are CCCs of Σ in $G[S']$, $G[S'']$, $\alpha' + \alpha'' = \alpha$ is the CCC of Σ in $G[S]$. Similarly, $\beta' + \beta'' = \beta$ is the CCC of $\bar{\Sigma}$ in $G[\bar{S}]$. As before, if $\pi = (\pi' - \|\beta'\|) + (\pi'' - \|\beta''\|) + \|\beta\|$ and $p = (p' - |\alpha'|) + (p'' - |\alpha''|) + |\alpha|$, then S is a feasible solution to $\Pi_i(\Sigma, \alpha, \beta, \pi, p)$, f_i is finite and

$$\begin{aligned} f_i(\Sigma, \alpha, \beta, \pi, p) \leq & k(S) = k(S') + k(S'') - k(\Sigma) = \\ & \min\Big\{f_{i'}(\Sigma, \alpha', \beta', \pi', p') + f_{i''}(\Sigma, \alpha'', \beta'', \pi'', p'') - k(\Sigma)\Big\}. \end{aligned}$$

The derived inequalities prove that equality (7) holds.

Appendix B Extensions to the Connected *MinMaxC* CNP

Many results obtained for the Connected Pairwise CNP can be extended to the *MinMaxC* CNP as well. We extend here the analysis of the complexity over graphs with bounded treewidth, trees, and series-parallel graphs.

B.1 NP-hardness on biconnected planar bipartite graphs

We can use the reasonings in the proof of Proposition 1 to show that the unweighted Connected *MinMaxC* CNP provides a solution to the MinCVC when the maximum cardinality of the remaining components must be one at most. We state the following proposition.

Proposition 7: The decision version of the Connected *MinMaxC* CNP is strongly NP-complete in biconnected planar bipartite graphs of maximum degree 4, even with unit node weights and unit deletion costs.

B.2 Graphs with bounded treewidth

We can modify the DP algorithm of Section 3 to prove similar results for the Connected *MinMaxC* CNP where the objective is to minimise the weight of the heaviest component in $G[\bar{S}]$. In order to

deal with such an objective function, we redefine the second elements of the pairs of a CCC α (or β). More precisely, we now define $\alpha = \{(\Sigma \cap V(H), c(V(H) \setminus \Sigma))\colon H \in \text{comp}(G), \Sigma \cap V(H) \neq \emptyset\}$ as the CCC of a set Σ in a graph G and we modify only the restriction operation of a potential CCC $\alpha = \{(A_i, a_i)\}_{i=1}^{k} \in \Gamma(\Sigma, r)$ as follows:

$$\alpha - v = \{(A_i \setminus \{v\}, a_i + c(A_i \cap \{v\}))\colon (A_i, a_i) \in \alpha,\ |A_i \setminus \{v\}| > 0\}.$$

Also, we now have $\|\beta\| = \max\{c(B_i) + b_i\}_{i=1\ldots k}$ for a CCC $\beta = \{(B_i, b_i)\}_{i=1\ldots k}$.

We consider the same function $f_i(\Sigma, \alpha, \beta, \pi, p)$ as in Section 3.3 but now $\pi = \max\{c(H) : H \in \text{comp}(G[\bar{S}])\}$. The dynamic program follows the same reasonings as those of the algorithm for the Connected Pairwise CNP. For a start bag $X_i = \{v\}$ we have

$$f_i(\Sigma, \alpha, \beta, \pi, p) = \begin{cases} k_v & \text{if } \Sigma = \{v\},\ \alpha = \{(\{v\}, 0)\},\ \beta = \emptyset,\ \pi = 0,\ p = 1 \\ 0 & \text{if } \Sigma = \emptyset,\ \alpha = \emptyset,\ \beta = \{(\{v\}, 0)\},\ \pi = c_v,\ p = 0 \\ +\infty & \text{in all other cases.} \end{cases}$$

The recursive functions (5)-(7) are respectively replaced with:

$$f_i(\Sigma, \alpha, \beta, \pi, p) = \min\Big\{\{f_j(\Sigma \cup \{v\}, \alpha', \beta, \pi, p)\colon \alpha' - v = \alpha\} \cup \{f_j(\Sigma, \alpha, \beta', \pi, p)\colon \beta' - v = \beta\}\Big\} \quad (15)$$

$$f_i(\Sigma, \alpha, \beta, \pi, p) = \begin{cases} \min\Big\{f_j(\Sigma \setminus \{v\}, \alpha', \beta, \pi, p')\colon & \\ \quad \alpha' + v = \alpha,\ p = p' + |\alpha| - |\alpha'|\Big\} + k_v & (v \in \Sigma) \\ \min\Big\{f_j(\Sigma, \alpha, \beta', \pi', p)\colon & \\ \quad \beta' + v = \beta,\ \pi = \max\{\pi', c_v + \sum_{l=1\ldots|\beta|:B_l \cap N(v) \neq \emptyset} (c(B_l) + b_l)\}\Big\} & (v \notin \Sigma) \end{cases} \quad (16)$$

$$f_i(\Sigma, \alpha, \beta, \pi, p) = \min\Big\{f_{i'}(\Sigma, \alpha', \beta', \pi', p') + f_{i''}(\Sigma, \alpha'', \beta'', \pi'', p'') - k(\Sigma)\colon \alpha' + \alpha'' = \alpha,\ \beta' + \beta'' = \beta,\ p' + p'' - |\alpha'| - |\alpha''| + |\alpha| = p,\ \max\{\pi', \pi'', \|\beta\|\} = \pi\Big\} \quad (17)$$

The optimal value is given by the state $f_1(\Sigma, \alpha, \beta, \pi, p) \leq K$ with $\alpha = \{(A_1, a_1)\}$ or $\alpha = \emptyset$, $p = 1$ and π as small as possible. The proof of correctness of our algorithm is very similar to that for the Connected Pairwise CNP. As regards the time complexity of the DP algorithm, given the definition of the CCCs we now have a number of indices in $f_i(\Sigma, \alpha, \beta, \pi, p)$ and a number of operations to perform for the recursive functions which are pseudo-polynomial in n and $c(V)$. We derive the following proposition and corollary.

Proposition 8: The Connected *MinMaxC* CNP with arbitrary node weights and arbitrary deletion costs can be solved in pseudo-polynomial time over graphs with a treewidth bounded by a constant κ.

Corollary 3: The Connected *MinMaxC* CNP with unit node weights and arbitrary deletion costs can be solved in polynomial time over graphs with a treewidth bounded by a constant κ.

We observe that Proposition 8 is more general than Theorem 1 and implies that over graphs with bounded treewidth, the Connected *MinMaxC* CNP is never strongly NP-hard. Besides, we will show in the next section that the Connected *MinMaxC* CNP over trees turns out to be polynomial even with arbitrary node weights and arbitrary deletion costs.

B.3 Complexity results over trees

The complexity of the *MinMaxC* CNP over trees has been studied in [40], where the problem was proved to be polynomial with unit node weights ($k_i = 1 \; \forall i \in \mathcal{T}$) and weakly NP-hard with arbitrary node weights and arbitrary deletion costs. We show here that the Connected *MinMaxC* CNP with arbitrary node weights and arbitrary deletion costs is a polynomial problem. We again consider all the subtrees where the root node is removed. For each $\mathcal{T}_a$, we solve to optimality the problem over the considered subtree by the following greedy strategy. After the removal of the root node a, we progressively remove the node i with maximum $c(\mathcal{T}_i)$ and whose father was already removed (in order to guarantee the connectivity of the deleted set), and iterate as long as the budget constraint is not violated. Notice that each value $c(\mathcal{T}_i)$ can be pre-computed in $\mathcal{O}(n)$. It is easy to see that this algorithm is optimal. Consider the initial situation where the root node is removed. Clearly, it would be suboptimal to remove other nodes (one or more) instead of the child node i with maximum $c(\mathcal{T}_i)$ as this operation would consume deletion budget without improving the objective function. So an optimal solution will indeed remove the node i whenever possible. The same argument applies for the progressive selection of the next nodes to remove in an optimal solution.

After solving the subproblem associated with $\mathcal{T}_a$, we take the maximum between the corresponding optimal value and $c(\mathcal{T} \setminus \mathcal{T}_a)$ to compute the value of the objective function over $\mathcal{T}$. An optimal solution for $\mathcal{T}$ is given by analyzing all the relevant subtrees and considering the subproblem that gives the minimum global objective function. The following proposition holds.

Proposition 9: The Connected *MinMaxC* CNP over trees admits a polynomial time algorithm with time complexity $\mathcal{O}(n^3)$.

Proof. The execution time of the algorithm for solving each problem associated with a subtree $\mathcal{T}_a$ is trivially bounded by $\mathcal{O}(n_a^2)$ since, at each iteration, we can select the node to remove in $\mathcal{O}(n_a)$ and n_a nodes are removed at most. Since the number of subtrees to consider is in $\mathcal{O}(n)$, the resulting time complexity is $\mathcal{O}(n^3)$. □

We summarise our results in Table 3.

B.4 Series-parallel graphs

We can state the following proposition for the Connected *MinMaxC* CNP over series-parallel graphs.

c_i	k_i	complexity	
		Connected *MinMaxC* CNP	*MinMaxC* CNP
≥ 0	> 0	Solvable in $\mathcal{O}(n^3)$	Weakly NP-hard [40]
$= 1$	> 0	Solvable in $\mathcal{O}(n^3)$	Solvable in $\mathcal{O}(n^3 \log n)$ [40]

Table 3: Summary of our complexity results for the Connected *MinMaxC* CNP over trees and comparison with the complexity results in [40] for the *MinMaxC* CNP over trees.

Theorem 5: The decision version of the Connected *MinMaxC* CNP over series-parallel graphs is NP-complete with arbitrary connection and deletion costs.

Proof. We can show the above theorem by performing a reduction from the Partition Problem, very similar to the one in Theorem 3. The only difference is that we give a node weight $c_i = a_i$ to all nodes $i \in A$, a deletion budget $K = \sum_{i \in A} a_i/2$ and a deletion cost $k_{n+1} = K + 1$ to node $n + 1$ so that it cannot be deleted anymore. We refer to Figure 4 for a visual example of the Connected *MinMaxC* CNP instance obtained. Consequently, the undeleted nodes will always form a single connected component whose total weight will be: $\sum_{i \in A: v_i^* = 0} a_i$. The rest of the proof is very similar to the proof of Theorem 3. □

Corollary 4: The Connected *MinMaxC* CNP is weakly NP-hard on series-parallel graphs with arbitrary node weights and deletion costs.

Proof. The DP algorithm of Section 3 for the Connected *MinMaxC* CNP is valid for integer node weights, which provides a pseudopolynomial time algorithm. Since the problem has an NP-complete decision version, the above result follows. □

We can use a slightly modified version of the above proof to show that the classic *MinMaxC* CNP on series-parallel graphs is NP-complete with arbitrary weights, which had not been investigated before.

Theorem 6: The decision version of the *MinMaxC* CNP is NP-complete on series-parallel graphs with arbitrary node weights and deletion costs.

Proof. We can perform the same reduction as in the above proof for the Connected *MinMaxC* CNP, changing the deletion weight of node $n + 1$ to $k_{n+1} = K + 1$ so it cannot be deleted and choosing $K = W$. The rest of the proof follows very similarly to the one of Theorem 6. □

Using the results of [40] we can formulate the following corollary:

Corollary 5: The *MinMaxC* CNP is Weakly NP-hard on series-parallel graphs with arbitrary node weights and deletion costs.

Proof. The recursion functions and equations of [40] can be easily extended to integer node weights, which provides a pseudopolynomial time algorithm. Since the problem has an NP-complete decision version, the above result follows. □

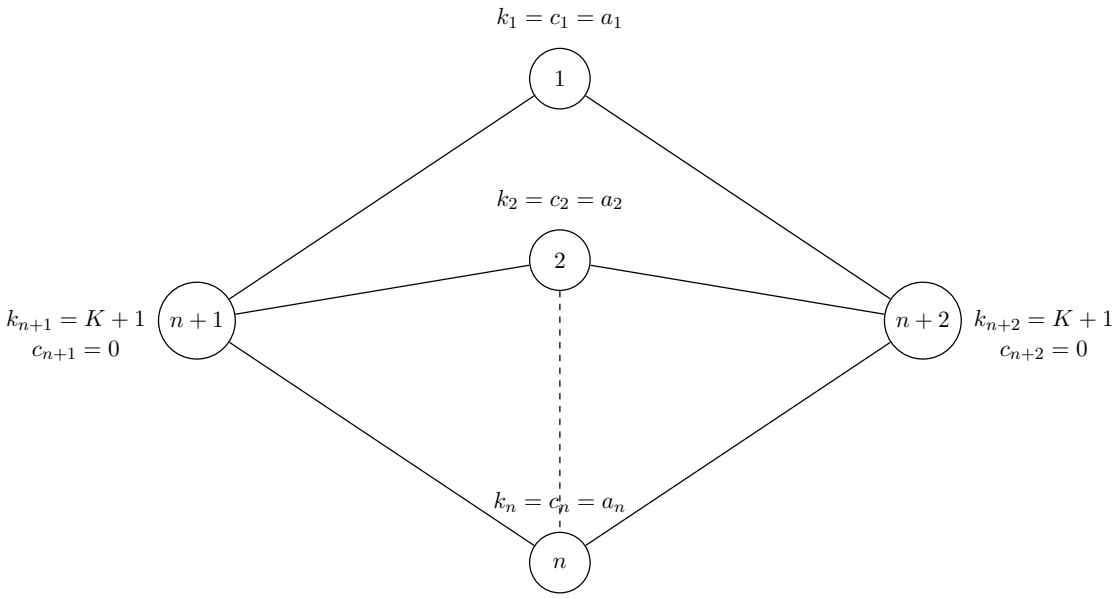

Figure 4: Example of a series-parallel instance for the Connected *MinMaxC* CNP over series-parallel graphs obtained by reduction from a Partition instance.

Our results for the Connected *MinMaxC* CNP over series-parallel graphs are summed up in Table 4.

c_i	k_i	complexity	
		Connected *MinMaxC* CNP	*MinMaxC* CNP
≥ 0	> 0	Weakly NP-hard	Weakly NP-hard
$= 1$	> 0	Polynomial	Polynomial [40]

Table 4: Summary of our complexity results for the Connected *MinMaxC* CNP based on the heaviest component weight over series-parallel graphs, along with complexity results for the CNP (partially from [40]).

Appendix C Extensions to the Connected *MaxNum* CNP

In this section, we present the results for the Connected *MaxNum* CNP. We first show the NP-hardness of the problem on biconnected planar bipartite graphs. Then, we illustrate our results on graphs with bounded treewidth. No additional meaningful findings can be derived for the Connected *MaxNum* CNP over trees or series-parallel graphs with respect to the more general results obtained in graphs with bounded treewidth.

C.1 NP-hardness on biconnected planar bipartite graphs

We can state the following proposition for the Connected *MaxNum* CNP:

Proposition 10: The decision version of the Connected *MaxNum* CNP is strongly NP-complete in biconnected planar bipartite graphs of maximum degree 4, even with unit deletion costs.

Proof. Analogously to the proof of Proposition 1, it suffices to reduce the decision version of the Minimum Connected Vertex Cover problem to the decision version of Connected *MaxNum* CNP with unit deletion costs. Given a connected graph G and a number K, the decision version of the MinCVC asks whether there exists a connected set S that is a vertex cover of G and such that $|S| \leq K$. Given an instance of the MinCVC, we can straightforwardly construct an instance of the Connected *MaxNum* CNP with unit deletion costs, a deletion budget K and a minimum threshold $n-K$ on the number of connected components. It is easy to check that if a Yes instance exists for the MinCVC with a solution S^* with, $|S^*| \leq K$, the number of connected components in $G[V \setminus S^*]$ is $n-|S^*| \geq n-K$. Hence, set S^* constitutes also a solution of the corresponding instance of the Connected *MaxNum* CNP. Conversely, consider a Yes instance of the Connected *MaxNum* CNP with solution S^* that induces a remaining graph with at least $n-K$ components. If $|S^*| = K$, clearly $|S^*|$ is also a connected vertex cover as each of the $n-K$ components in $G[V \setminus S^*]$ must have one node only. Suppose now that $|S^*| < K$. Since each component in $G[V \setminus S^*]$ must necessarily have at least one neighbour in S^*, we can iteratively add nodes in S^* by adding nodes from the connected components in $G[V \setminus S^*]$ (with cardinality > 1) until again $|S^*| = K$ and ensuring that set S^* is connected. Thus, a Yes instance of the Connected *MaxNum* CNP implies a Yes instance of the MinCVC, completing the reduction. □

C.2 Graphs with bounded treewidth

We also extend the proposed DP algorithm of Section 3 to the Connected *MaxNum* CNP. Here we consider the definition of $f_i(S, \alpha, \beta, \pi, p)$ as in Section 3.3 but with $\pi = |\text{comp}(G[\bar{S}])|$ and do not need to modify the definition of a CCC. For a start bag $X_i = \{v\}$ we now have

$$f_i(\Sigma, \alpha, \beta, \pi, p) = \begin{cases} k_v & \text{if } \Sigma = \{v\},\ \alpha = \{(\{v\}, 0)\},\ \beta = \emptyset,\ \pi = 0,\ p = 1 \\ 0 & \text{if } \Sigma = \emptyset,\ \alpha = \emptyset,\ \beta = \{(\{v\}, 0)\},\ \pi = 1,\ p = 0 \\ +\infty & \text{in all other cases.} \end{cases}$$

We can then replace the recursive functions (5)-(7) respectively with:

$$f_i(\Sigma, \alpha, \beta, \pi, p) = \min \Big\{ \{ f_j(\Sigma \cup \{v\}, \alpha', \beta, \pi, p) \colon \alpha' - v = \alpha \} \cup \{ f_j(\Sigma, \alpha, \beta', \pi, p) \colon \beta' - v = \beta \} \Big\} \quad (18)$$

$$f_i(\Sigma, \alpha, \beta, \pi, p) = \begin{cases} \min \{ f_j(\Sigma \setminus \{v\}, \alpha', \beta, \pi, p') \colon \\ \qquad \alpha' + v = \alpha,\ p = p' + |\alpha| - |\alpha'| \} + k_v & (v \in \Sigma) \\ \min \{ f_j(\Sigma, \alpha, \beta', \pi', p) \colon \\ \qquad \beta' + v = \beta,\ \pi = \pi' + |\beta| - |\beta'| \} & (v \notin \Sigma) \end{cases} \quad (19)$$

$$\begin{aligned} f_i(\Sigma, \alpha, \beta, \pi, p) = \min \Big\{ & f_{i'}(\Sigma, \alpha', \beta', \pi', p') + f_{i''}(\Sigma, \alpha'', \beta'', \pi'', p'') - k(\Sigma) \colon \\ & \alpha' + \alpha'' = \alpha,\ \beta' + \beta'' = \beta, \\ & p' + p'' - |\alpha'| - |\alpha''| + |\alpha| = p, \\ & \pi' + \pi'' - |\beta'| - |\beta''| + |\beta| = \pi \Big\} \end{aligned} \quad (20)$$

The optimal value is given by the state $f_1(\Sigma, \alpha, \beta, \pi, p) \leq K$ with $\alpha = \{(A_1, a_1)\}$ or $\alpha = \emptyset$, $p = 1$ and π as large as possible. Again, the proof of correctness of the algorithm is very similar to that for the Connected Pairwise CNP.

Proposition 11: The Connected *MaxNum* CNP with arbitrary deletion costs can be solved in polynomial time over graphs with a treewidth bounded by a constant κ.

References

[1] B. Addis, R. Aringhieri, A. Grosso, and P. Hosteins. Hybrid Constructive Heuristics for the Critical Node Problem. *Annals of Operations Research*, 238(1):637–649, 2016.

[2] B. Addis, M. Di Summa, and A. Grosso. Removing critical nodes from a graph: complexity results and polynomial algorithms for the case of bounded treewidth. *Discrete Applied Mathematics*, 16-17:2349–2360, 2013.

[3] R. Albert, H. Jeong, and A. L. Barabási. Error and attack tolerance of complex networks. *Nature*, 406:378–382, 2000.

[4] D. Alevras, M. Grötschel, and R. Wessäly. Capacity and survivability models for telecommunication networks. Technical report, in Proceedings of EURO/INFORMS Meeting, 1997.

[5] R. Aringhieri, A. Grosso, P. Hosteins, and R. Scatamacchia. A general Evolutionary Framework for different classes of Critical Node Problems. *Engineering Applications of Artificial Intelligence*, 55:128–145, 2016.

[6] R. Aringhieri, A. Grosso, P. Hosteins, and R. Scatamacchia. Local Search Metaheuristics for the Critical Node Problem. *Networks*, 67(3):209–221, 2016.

[7] R. Aringhieri, A. Grosso, P. Hosteins, and R. Scatamacchia. Polynomial and pseudo-polynomial time algorithms for different classes of the Distance Critical Node Problem. *Discrete Applied Mathematics*, 253:103–121, 2019.

[8] S. Arnborg, D. G. Corneil, and A. Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM Journal on Algebraic and Discrete Methods*, 8:277–284, 1987.

[9] A. Arulselvan, C. W. Commander, L. Elefteriadou, and P. M. Pardalos. Detecting critical nodes in sparse graphs. *Computers & Operations Research*, 36(7):2193–2200, 2009.

[10] A. Arulselvan, C. W. Commander, O. Shylo, and P. M. Pardalos. Cardinality-constrained critical node detection problem. In Nalân Gülpnar, Peter Harrison, and Berç Rüstem, editors, *Performance Models and Risk Management in Communications Systems*, volume 46 of *Springer Optimization and Its Applications*, pages 79–91. Springer New York, 2011.

[11] R.B. Bapat, S. Fujita, S. Legay, Y. Manoussakis, Y. Matsui, T. Sakuma, and Z. Tuza. Safe sets, network majority on weighted trees. *Networks*, 71(1):81–92, 2018.

[12] A. Berger, A. Grigoriev, and R. Zwaan. Complexity and approximability of the k-way vertex cut. *Networks*, 18(2):170–178, 2014.

[13] H. L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM Journal on Computing*, 25:1305–1317, 1996.

[14] S. P. Borgatti. Identifying sets of key players in a network. *Computational and Mathematical Organization Theory*, 12:21–34, 2006.

[15] G. Brown, M. Carlyle, J. Salmerón, and K. Wood. Defending critical infrastructure. *Interfaces*, 36(6):530–544, 2006.

[16] G. Cho and D. X. Shaw. A depth–first dynamic programming algorithm for the tree knapsack problem. *INFORMS Journal on Computing*, 9:431–438, 1997.

[17] R. Cohen, D. Ben-Avraham, and S. Havlin. Efficient immunization strategies for computer networks and populations. *Physical Review Letters*, 91:247901–247905, 2003.

[18] M. Di Summa, A. Grosso, and M. Locatelli. Complexity of the critical node problem over trees. *Computers and Operations Research*, 38:1766–1774, 2011.

[19] T. N. Dinh, Y. Xuan, M. T. Thai, E. K. Park, and T. Znati. On approximation of new optimization methods for assessing network vulnerability. In *Proceedings of the 29th IEEE Conference on Computer Communications (INFOCOM)*, pages 105–118, 2010.

[20] T.N. Dinh, Y. Shen, D.T. Nguyen, and M.T. Thai. On the approximability of positive influence dominating set in social networks. *Journal of Combinatorial Optimization*, 27:487–503, 2014.

[21] B. Escoffier, L. Gourvès, and J. Monnot. Complexity and approximation results for the connected vertex cover problem in graphs and hypergraphs. *Journal of Discrete Algorithms*, 8:36–49, 2010.

[22] S. Fujita, G. MacGillivray, and T. Sakuma. Safe set problem on graphs. *Discrete Applied Mathematics*, 215:106–111, 2016.

[23] M.R. Garey and D.S. Johnson. *Computers and intractability: A guide to the theory of NP-completeness*. W. H. Freeman and Company, San Francisco, 1979.

[24] N. Garg, V.V. Vazirani, and M. Yannakakis. Primal-dual approximation algorithms for integral flow and multicut in trees. *Algorithmica*, 18:3–30, 1997.

[25] D. Granata, G. Steeger, and S. Rebennack. Network interdiction via a critical disruption path: Branch-and-price algorithms. *Computers & Operations Research*, 40(11):2689 – 2702, 2013.

[26] T. H. Grubesic and A. T. Murray. Vital nodes, interconnected infrastructures, and the geographies of network survivability. *Ann Association American Geographers*, 96:64–83, 2006.

[27] P. Hosteins and R. Scatamacchia. The Stochastic Critical Node Problem over trees. *Networks*, 76(3):381–401, 2020.

[28] E. Jenelius, T. Petersen, and L.-G. Mattsson. Importance and exposure in road network vulnerability analysis. *Transportation Research Part A: Policy and Practice*, 40(7):537 – 560, 2006.

[29] D. S. Johnson and K. A. Niemi. On knapsacks, partitions, and a new dynamic programming technique for trees. *Mathematics of Operations Research*, 8:1–14, 1983.

[30] T. Kloks. *Treewidth: Computations and Approximations*, volume 842 of *Lecture Notes in Computer Science*. Springer, 1994.

[31] M. Lalou and H. Kheddouci. A polynomial-time algorithm for finding critical nodes in bipartite permutation graphs. *Optimization Letters*, 13:1345–1364, 2019.

[32] M. Lalou, M. A. Tahraoui, and H. Kheddouci. Component-cardinality-constrained critical node problem in graphs. *Discrete Applied Mathematics*, 210:150–163, 2016.

[33] M. Lalou, M. A. Tahraoui, and H. Kheddouci. The critical node detection problem in networks: A survey. *Computer Science Review*, 28:92–117, 2018.

[34] C. Lim and J. C. Smith. Algorithms for discrete and continuous multicommodity flow network interdiction problems. *IIE Transactions*, 39:15–26, 2007.

[35] G. Lin, J. Guan, and H. Feng. An ilp based memetic algorithm for finding minimum positive influence dominating sets in social networks. *Physica A: Statistical Mechanics and its Applications*, 500:199–209, 2018.

[36] T. C. Matisziw and A. T. Murray. Modeling s-t path availability to support disaster vulnerability assessment of network infrastructure. *Computers & Operations Research*, 36:16–26, 2009.

[37] W. Pullan. Heuristic identification of critical nodes in sparse real-world graphs. *Journal of Heuristics*, 21(5):577–598, 2015.

[38] D. Purevsuren, G. Cui, N. N. Htay Win, and X. Wang. Heuristic algorithm for identifying critical nodes in graphs. *Advances in Computer Science : an International Journal*, 5(3):1–4, 2016.

[39] J. Salmerón, K. Wood, and R. Baldick. Analysis of electric grid security under terrorist threat. *IEEE Trans Power Syst*, 19:905–912, 2004.

[40] S. Shen and J. C. Smith. Polynomial-time algorithms for solving a class of critical node problems on trees and series-parallel graphs. *Networks*, 60(2):103–119, 2012.

[41] S. Shen, J. C. Smith, and R. Goli. Exact interdiction models and algorithms for disconnecting networks via node deletions. *Discrete Optimization*, 9:172–188, 2012.

[42] J. C. Smith, M. Prince, and J. Geunes. Modern network interdiction problems and algorithms. In P. M. Pardalos, D.-Z. Du, and R. L. Graham, editors, *Handbook of Combinatorial Optimization*, pages 1949–1987. Springer New York, New York, NY, 2013.

[43] K. Takamizawa, T. Nishizeki, and N. Saito. Linear-time computability of combinatorial problems on series-parallel graphs. *Journal of the ACM*, 29:623–641, 1982.

[44] J. Valdes, R.E. Tarjan, , and E.L. Lawler. The recognition of series-parallel digraphs. *SIAM Journal on Computing*, 11:298–313, 1982.

[45] M. Ventresca. Global search algorithms using a combinatorial unranking-based problem representation for the critical node detection problem. *Computers & Operations Research*, 39(11):2763–2775, 2012.

[46] M. Ventresca and D. Aleman. Efficiently identifying critical nodes in large complex networks. *Computational Social Networks*, 2(6), 2015.

[47] A. Veremyev, O. A. Prokopyev, and E. L. Pasiliao. An integer programming framework for critical elements detection in graphs. *Journal of Combinatorial Optimization*, 28:233–273, 2014.

[48] A. Veremyev, O. A. Prokopyev, and E. L. Pasiliao. Critical nodes for distance-based connectivity and related problems in graphs. *Networks*, 66:170–195, 2015.

[49] J. L. Walteros, A. Veremyev, P. M. Pardalos, and E. L. Pasiliao. Detecting critical node structures on graphs: A mathematical programming approach. *Networks*, 2018.

[50] R. K. Wood. Deterministic network interdiction. *Mathematical and Computer Modelling*, 17:1–18, 1993.

[51] T. Zhou, Z. Q. Fu, and B. H. Wang. Epidemic dynamics on complex networks. *Progress in Natural Sciences*, 16:452–457, 2006.