Parameterized Complexity of Critical Node Cuts

Danny Hermelin*[1], Moshe Kaspi[1], Christian Komusiewicz[2], and Barak Navon[1]

[1] Department of Industrial Engineering and Management, Ben-Gurion University, Israel
hermelin@bgu.ac.il, moshe@exchange.bgu.ac.il, baraknav@post.bgu.ac.il
[2] Institut für Softwaretechnik und Theoretische Informatik, TU Berlin, Germany
christian.komusiewicz@tu-berlin.de

Abstract. We consider the following natural graph cut problem called Critical Node Cut (CNC): Given a graph G on n vertices, and two positive integers k and x, determine whether G has a set of k vertices whose removal leaves G with at most x connected pairs of vertices. We analyze this problem in the framework of parameterized complexity. That is, we are interested in whether or not this problem is solvable in $f(\kappa) \cdot n^{O(1)}$ time (*i.e.*, whether or not it is *fixed-parameter tractable*), for various natural parameters κ. We consider four such parameters:

- The size k of the required cut.
- The upper bound x on the number of remaining connected pairs.
- The lower bound y on the number of connected pairs to be removed.
- The treewidth w of G.

We determine whether or not CNC is fixed-parameter tractable for each of these parameters. We determine this also for all possible aggregations of these four parameters, apart from $w + k$. Moreover, we also determine whether or not CNC admits a polynomial kernel for all these parameterizations. That is, whether or not there is an algorithm that reduces each instance of CNC in polynomial time to an equivalent instance of size $\kappa^{O(1)}$, where κ is the given parameter.

1 Introduction

In 2013 a polio virus struck Israel. The virus spread in alarming speed, creating a nationwide panic of parents concerned about the well-being of their children. It was obvious to the Israeli health department that vaccinating all Israeli children is not a practical solution in the given time frame. Thus it became clear that some areas of the population should be vaccinated first in order to stop the spread of the virus as quickly as possible. Let us represent a geographic area as a vertex of a graph, and the roads between areas as edges of the graph. In this setting, vaccinating an area corresponds to deleting a certain vertex from the graph. Thus, the objective of stopping the virus from spreading translates to minimizing the number of *connected pairs* (two vertices which are in the same component) in the corresponding graph after applying the vaccination.

This scenario can be modeled by the following graph-theoretic problem which we call Critical Node Cut (CNC). In this problem, we are given an undirected simple graph G and two integers k and x. The objective is to determine whether there exists a set $C \subseteq V(G)$ of at most k vertices in G, such that the graph $G - C$ which results from removing C from G contains at most x connected pairs. In this sense, the cut C is considered *critical* since removing it from G leaves few (at most x) connected pairs. For convenience purposes, throughout the paper we will count *ordered* connected pairs; *i.e.*, pairs $(u, v) \in V(G) \times V(G)$, $u \neq v$, where u and v belong to same connected component in $G - C$.

The goal of CNC is thus, roughly speaking, to destroy the connectivity of a given graph as much as possible given a certain budget for deleting vertices. From this point of view, CNC fits nicely to the broad family of *graph cut problems*. Many problems from this have been studied widely and are among the most fundamental problems in algorithmic research, including Min Cut, Max Cut, Multicut, Multiway Cut, Feedback Vertex Set, and Vertex Cover (see *e.g.* [15] for definitions of these problems). The latter

* The research leading to these results has received funding from the People Programme (Marie Curie Actions) of the European Union's Seventh Framework Programme (FP7/2007-2013) under REA grant agreement number 631163.11, and by the ISRAEL SCIENCE FOUNDATION (grant No. 551145/).

Parameter				Result	
k	x	y	w	FPT	P-Kernel
✓				NO (Thm. 1)	NO (Thm. 1)
	✓			NO	NO
		✓		YES (Thm. 6)	NO (Thm. 7)
			✓	NO (Thm. 4)	NO (Thm. 7)
✓	✓			YES (Thm. 2)	YES (Thm. 3)
✓		✓		YES (Thm. 6)	NO (Thm. 7)
✓			✓	**?**	NO (Thm. 7)
	✓	✓		YES	YES
	✓		✓	YES (Thm. 5)	NO
		✓	✓	YES (Thm. 6)	NO (Thm. 7)
✓	✓	✓		YES (Thm. 2)	YES (Thm. 3)
✓	✓		✓	YES (Thm. 2)	YES (Thm. 3)
✓		✓	✓	YES (Thm. 6)	NO (Thm. 7)
	✓	✓	✓	YES	YES
✓	✓	✓	✓	YES	YES

Table 1: Summary of the complexity results for Critical Node Cut.

is the special case of CNC with $x = 0$. Since Vertex Cover is arguably the most important problem in the theory of algorithmic design for NP-hard problems, CNC provides a natural test bed to see which of the techniques from this theory can be extended, and to what extent.

Related Work and Applications. The CNC problem has been studied from various different angles. The problem was shown to be NP-complete in [2] (although its NP-completeness follows directly from the much earlier NP-completeness result for Vertex Cover). In trees, a weighted version of CNC is NP-complete whereas the non-weighted version can be solved in polynomial time [10]. The case of bounded treewidth can be solved using dynamic programming in $O(n^{w+1})$ time, where n is the number of vertices in the graph and w is its treewidth [1]. Local search [2] and simulated annealing [20] were proposed as heuristic algorithms for CNC. Finally, in [21] an approximation algorithm based on randomized rounding was developed.

Due to its generic nature, the CNC problem has been considered above in various different application settings. One example application is the virus vaccination problem discussed above [2]. Other interesting applications include protecting a computer/communication network from corrupted nodes, analyzing anti-terrorism networks [18], measuring centrality in brain networks [17], insulin signaling [19], and protein-protein interaction network analysis [6].

Our Results. From reviewing the literature mentioned above, it is noticeable that an analysis of CNC from the perspective of parameterized complexity [11] is lacking. The purpose of this paper is to remedy this situation. We examine CNC with respect to four natural parameters along with all their possible combined aggregations. The four basic parameters we examine are:

- The size k of the solution (*i.e.* the critical node cut) C.
- The bound x on the number of connected pairs in the resulting graph $G - C$.
- The number of connected pairs y to be removed from G; if G is connected with n vertices then $y = n(n-1) - x$.
- The treewidth w of G.

Table 1 summarizes all we know regarding the complexity of CNC with respect to these four parameters and their aggregation. Let us briefly go through some of the trivial results given in the table above. First note that CNC with $x = 0$ is precisely the Vertex Cover problem, which means that CNC is not in FPT (and therefore has no polynomial kernel) for parameter x unless P=NP. This also implies that the problem

is unlikely to admit a polynomial kernel even when parameterized by $w+x$, since such a kernel would imply a polynomial kernel for Vertex Cover parameterized by the treewidth w which is known to cause the collapse of the polynomial hierarchy [5,12]. Next, notice that if our input graph G has no isolated vertices, we have $x+y=\Omega(n)$, and therefore CNC is FPT and has a polynomial kernel (as isolated vertices can safely be discarded). This of course means that the same applies for parameters $k+x+y$, $x+y+w$, and $k+x+y+w$.

Our first result, stated in Theorem 1, shows that CNC parameterized by k is W[1]-hard. Thus, CNC is unlikely to have an FPT algorithm under this parameterization. We then show in Theorem 2 and Theorem 3, that when considering $x+k$ as a parameter, we can extend two classical Vertex Cover techniques to the CNC problem. Our main technical result is stated in Theorem 4, where we prove that CNC is W[1]-hard with respect to w, the treewidth of the input graph. This is somewhat surprising since not many graph cut problems are known to be W[1]-hard when parameterized by treewidth. Also, the result complements nicely the $O(n^{w+1})$-time algorithm of [1] by showing that this algorithm cannot be improved substantially. (Specifically, our result implies that there is no $n^{o(\sqrt[6]{w})}$-algorithm for CNC unless the so-called Exponential Time Hypothesis [16] fails). We complement this algorithm from the other direction by showing in Theorem 5 that CNC can be solved in $w^{O(w+x)}\cdot n^{O(1)}$ time. Finally, we show in Theorem 6 and Theorem 7 that CNC is FPT with respect to y, and has no polynomial kernel even if y, w, and k are taken together as parameters.

2 Parameters k and $k+x$

We now consider the parameters k and $k+x$ for CNC. We will show that the problem is W[1]-hard for the former parameterization, while for the latter it is in FPT and admits a polynomial kernel.

Theorem 1. Critical Node Cut *is* W[1]*-hard with respect to* k.

Proof. We present a reduction from the Clique problem: Given a graph G on n vertices, and a parameter ℓ, determine whether G has a pairwise adjacent subset of ℓ vertices. Let (G,ℓ) be an instance of Clique. We construct H, the graph of our CNC instance, as follows: We replace each edge in G by a simple *edge-gadget*. This is done by replacing the edge by n parallel edges, and then subdividing each of the new edges once. The newly inserted subdivision vertices are referred to as *dummy vertices*. We then add an edge in H between each pair of nonadjacent vertices of G. Finally, we set $k:=\ell$.

We claim that the graph G has a clique of size ℓ if and only if the graph H has $k=\ell$ vertices whose removal deletes $y=\binom{k}{2}(N-1)n+k(N-1)$ connected pairs in H, where $N:=|V(H)|$. We begin with the easier direction: Suppose C is a clique of size ℓ in G. Then an easy calculation shows that removing C in H results in the deletion of y connected pairs from H: A total of $k(N-1)$ connected pairs which involve vertices of C, and $\binom{k}{2}(N-1)n$ connected pairs that involve dummy vertices adjacent to vertices in C.

Conversely, suppose that C is a cut that removes y connected pairs in H. Observe that if C contains a subset $C'\subseteq C$ of dummy vertices, then we can replace C' with an arbitrary equally sized set of non-dummy vertices without decreasing y. Thus, we can assume that C contains only non-dummy vertices. Furthermore, notice that when we remove a non-dummy vertex v (*i.e.*, a vertex of G), then the only connected pairs that are deleted are the ones which either involve v or possibly dummy vertices that are neighbors of v. This is because every pair of vertices from G is either connected by an edge or by an edge-gadget in H. Thus, the only way to delete $\binom{k}{2}(N-1)n+k(N-1)$ connected pairs in H is to delete k vertices which are pairwise connected by edge-gadgets; these consequently correspond to $\ell=k$ vertices that form a clique in G. □

We next show that the above result holds also for some restricted subclasses. A *split graph* is a graph in which the vertices can be partitioned into a clique and an independent set. We slightly modify the construction in the proof of Theorem 1 by adding all the edges missing between every pair of non-dummy vertices. In this way, the vertices of G form a clique and the dummy vertices form an independent set, while all arguments in the proof above still hold. For a fixed integer $d\geq 1$, a graph is called *d-degenerate* if each of its subgraphs has a vertex with a degree of at most d. For $d=1$ (*i.e.* a forest), the CNC problem has a polynomial algorithm based on dynamic programming [10]. We modify the construction in the proof above

by subdividing all the edges except those that are adjacent to dummy vertices. This results in a 2-degenerate graph, and also a bipartite graph with one side containing all vertices of G and the other containing all the dummy vertices. By a slightly more careful (yet still on the same lines) argument it can be shown that the conclusion of Theorem 1 still stands.

Corollary 1. Critical Node Cut *remains* W[1]*-hard with respect to* k *even if the input graph is split, bipartite, or* d*-degenerate for any fixed* $d \geq 2$*.*

We next consider the parameter $k + x$. We will show that the basic techniques known for the case of $x = 0$, *i.e.*, the Vertex Cover problem, can be extended to the case where $x > 0$. First, a simple branching strategy can be developed into an FPT algorithm for the parameter $k + x$.

Theorem 2. Critical Node Cut *is* FPT *with respect to* $k + x$*.*

Proof. Let (G, k, x) be an instance of CNC, and let n denote the number of vertices in G. Observe that if there exists a $C \subseteq V(G)$ of size k such that $G - C$ has at most x connected pairs, then $G - C$ has at most x edges. Using this observation we will solve an auxiliary problem in order to determine whether (G, k, x) has a solution. The objective of our auxiliary problem is to determine whether there exist k vertices $C' \subseteq V(G)$ such that $G - C'$ has at most x edges. Observe that we can solve this problem using the bounded search tree technique. For an arbitrary edge $\{u, v\} \in G$, we recursively branch on each of the following instances $(G - u, k - 1, x)$, $(G - v, k - 1, x)$, and $(G - \{u, v\}, k, x - 1)$. This process continues recursively until no edges remain, or $k = 0$, or $x = 0$.

An important attribute of this search tree algorithm is that it enumerates all the possible minimal solutions. Therefore, after applying the above algorithm, we obtain the set $\mathcal{C}'$ of all the minimal solutions to our auxiliary problem. If there exists a solution C to our CNC instance, then C is also a solution for the auxiliary problem but not necessarily a minimal solution. We apply brute force on each minimal solution in $\mathcal{C}'$ to check if it is possible to extend it into a solution for CNC. If this is not possible for every solution in $\mathcal{C}'$, then our CNC instance (G, k, x) has no solution.

To analyze the running time of the algorithm described above, note that solving our auxiliary problem requires $3^{x+k} \cdot n$ time, and the size of the set of all minimal solutions $\mathcal{C}'$ generated by this algorithm is bounded by 3^{x+k}. Let us next bound the running-time required for processing each minimal solution: Assume $C' \in \mathcal{C}'$ contains k_1 vertices, leaving us a budget of $k_2 = k - k_1$ vertices for our critical node cut. Now we can discard isolated vertices in $G - C'$ since these are irrelevant, and obtain a graph with at most $2x$ vertices. If $2x > k_2$, then all vertices can be deleted. Otherwise, checking each possible way to extend C' into a critical node cut requires $O(\binom{2x}{k_2}x^2 + n)$ time, and the running time of the entire algorithm is $O(4^x \cdot 3^{k+x} \cdot x^2 n)$. □

We remark that the running time can be improved by using a more elaborate approach in the last step. For example, isolated edges can be dealt with in a dynamic programming subroutine. Then the remaining instance on which brute-force has to be applied has at most $3x/2$ vertices. Next, we show that a simple "high-degree rule" leads to a polynomial kernel.

Theorem 3. Critical Node Cut *has a polynomial kernel with respect to* $k + x$*.*

Proof. Let (G, k, x) be an instance of CNC. We will show a polynomial reduction from (G, k, x) to an equivalent instance (G', k', x) of CNC such that the number of vertices in G' is polynomial in $k + x$. Our algorithm is in the same spirit of Buss's classical Vertex cover kernel [7]. We construct G' by iteratively applying a high-degree rule until it can no longer be applied: Start with $k' = k$. Note that a vertex with more than $k' + x$ neighbors must be in any critical node cut of size k. Thus, our high-degree rule checks if there is a vertex with degree at least $k' + x + 1$ in the graph, and if so, it removes it and decreases k' by one. Once all high degree vertices are removed, we remove all isolated vertices to obtain G'. Clearly, this reduction runs in polynomial time. Moreover, (G, k, x) is a yes-instance iff (G', k', x).

Let us next bound the number of vertices in G'. Suppose there is a $C \subseteq V(G')$ of at most $k' \leq k$ vertices where $G' - C$ has at most x connected pairs. We partition the remaining vertices of G' into two sets A and B, $A \cup B = V(G) \setminus C$. The set A contains all isolated vertices in $G' - C$, while B contains the non-isolated

vertices of $G' - C$. Clearly $|B| \leq x$, since otherwise there would be more than x connected pairs in $G' - C$. Now, since G' has no isolated vertices by construction, each vertex of A is adjacent to at least one vertex of C. Moreover, since each vertex of C has at most $k + x$ neighbors, and C has at most k vertices, this implies that $|A| \leq k(k + x)$. Thus,

$$|V(G')| = |A| + |B| + |C| \leq k(k + x) + x + k = (k + 1)(x + k)$$

and the theorem is proved. □

3 Parameter w

In this section we will show that CNC is unlikely to be fixed-parameter tractable when parameterized by w. This implies that we cannot substantially improve on the $O(n^{w+1})$ algorithm of [1]. Since we will not directly use the notion of treewidth and tree decompositions, we will defer their definitions to Section 4. The main result of this section is stated below.

Theorem 4. Critical Node Cut *is* W[1]*-hard with respect to the treewidth w of the input graph.*

Our proof of the theorem above is via the well-known multicolored clique technique [13] which utilizes generic gadget structure to construct a reduction from the W[1]-complete Multicolored Clique problem: Given an undirected simple graph G with n vertices and m edges, a coloring function $c : V(G) \to \{1, \ldots, \ell\}$ of the vertices of G, and a parameter ℓ, determine whether G has a clique which includes exactly one vertex from each color. Throughout the section we use (G, c, ℓ) to denote an arbitrary input to Multicolored Clique. As usual in parameterized reductions, we can assume that n and ℓ are sufficiently larger than any fixed constant, and n is sufficiently larger than ℓ.

In the multicolored clique technique, we construct *selection* gadgets which encode the selection of vertices and edges of G (one per each color class and pair of color classes, respectively), and *validation* gadgets which ensure that the vertices and edges selected indeed form a clique in G. In our reduction below, we will force any feasible solution to delete a large number of vertices from the constructed CNC instance in order to reach the required bound on the number of remaining connected pairs. We will ensure that such a solution always leaves $4\binom{\ell}{2}$ very large components which encode the selection of $\binom{\ell}{2}$ edges in G. The bound on the number of connected pairs will require all these huge components to have equal size, which in turn can only happen if the edges selected in G are edges between the same set of ℓ vertices (implying that these ℓ vertices form a clique in G). In what follows, we use (H, k, x) to denote the instance of CNC that we construct, where H is the input graph, k is the size of the required cut, and x is the bound on the number of connected pairs. Note that for our proof to go through, we will also need to show that the treewidth of H is bounded by some function in ℓ.

Connector gadgets: To each vertex $u \in V(G)$, we assign two unique integer identifiers: $low(u) \in \{1, \ldots, n\}$ and $high(u) \in \{n+1, \ldots, 2n\}$, where $high(u) = 2n - low(u)$. Our selection gadgets are composed from gadgets which we call *connector gadgets*. A connector gadget *corresponds* to a vertex of G, and can be of *low order* or *high order*. A low order connector gadget corresponding to a vertex $u \in V(G)$ consists of a clique of size ℓ^2 and an independent set of size $n^5 + low(u)$ which have all edges between them; that is, it is a complete split graph on these two sets of vertices. Similarly, a high order connector gadget corresponding to $u \in V(G)$ is a complete split graph on a clique of size ℓ^2 and an independent set of size $n^5 + high(u)$.

We refer to the clique in a connector gadget as the *core* of the gadget, and to the remaining vertices as the *guard* of the gadget. Only vertices in the core will be adjacent to vertices outside the gadget. Notice that the huge independent set in the core contributes to a large number of connected pairs in H, and one can delete all these connected pairs only by adding all core vertices to the solution cut. Below we use this property to help us control solutions for our CNC instance.

Selection gadgets: The graph H consists of a selection gadget for each vertex and edge in G (see Figure 1): For a vertex $u \in V(G)$, we will construct a *u-selection gadget* as follows: First we add u to H, and then

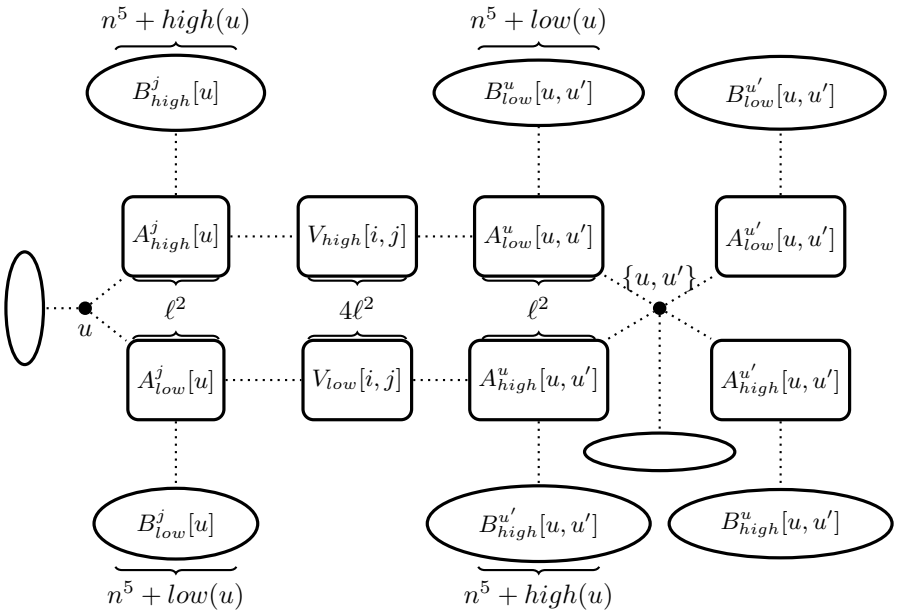

Fig. 1: The connection of selection gadgets via a validation gadget. In the example, we consider a vertex $u \in V(H)$ with $c(u) = i$ which is adjacent to a vertex $u' \in V(H)$ with $c(u') = j$. The diagram depicts the pair of low and high connector gadgets associated with color j in the u-selection gadget that are connected to the $\{u, u'\}$-selection gadget. The remaining $(k-2)$ pairs of connector gadgets in the u-selection gadget are not depicted. The rectangle boxes represent cliques and each ellipsis represents an independent set. The independent sets adjacent to u and $\{u, u'\}$ each have size n^2. The dotted lines depict a complete set of edges between two sets of vertices.

we connect u to $(\ell - 1)$ gadget pairs, one pair for each color $i \in \{1, \ldots, \ell\} \setminus \{c(u)\}$. Each pair consists of a low order and a high order connector gadget corresponding to u. We let $A^i_o[u]$ and $B^i_o[u]$ respectively denote the core and guard of the connector gadget *associated* with color $i \in \{1, \ldots, \ell\} \setminus \{c(u)\}$ and of order $o \in \{low, high\}$. We connect u to each connector gadget by adding all edges between $u \in V(H)$ and $A^i_o[u]$, for each $i \in \{1, \ldots, \ell\} \setminus \{c(u)\}$ and $o \in \{low, high\}$. We also connect u to an additional set of n^2 vertices of degree 1 in H, which we call the *dummy neighbors of* u.

For an edge $\{u_1, u_2\} \in E(G)$, we will construct a $\{u_1, u_2\}$-*selection gadget* as follows: First we add a vertex which we denote by $\{u_1, u_2\}$ to H. We then connect $\{u_1, u_2\} \in V(H)$ to a low order and a high order connector gadget associated with u_1, and to a low order and a high order connector gadget associated with u_2, by adding all edges between vertex $\{u_1, u_2\} \in V(H)$ and the core vertices of these gadgets. We let $A^u_o[u_1, u_2]$ and $B^u_o[u_1, u_2]$ respectively denote the core and guard of the connector gadget corresponding to $u \in \{u_1, u_2\}$ of order $o \in \{low, high\}$ in the $\{u_1, u_2\}$-selection gadget. Finally, we connect $\{u_1, u_2\} \in V(H)$ to an additional set of n^2 dummy neighbors of degree one in H.

Validation gadgets: We next add the validation gadgets to H, one for each ordered pair of distinct colors (i, j), $i \neq j$. For such a pair (i, j), the (i, j)-validation gadget simply consists of two cliques $V_{low}[i, j]$ and $V_{high}[i, j]$, each of size $4\ell^4$. The validation is done through the connections of these two cliques to the remainder of the graph. Consider a u-selection gadget for a vertex $u \in V(G)$ of color i. We add all possible edges between $V_{low}[i, j]$ and $A^j_{low}[u]$, and all edges between $V_{high}[i, j]$ and $A^j_{high}[u]$. This is done for every vertex of color i. Consider next a $\{u_1, u_2\}$-selection gadget where $c(u_1) = i$ and $c(u_2) = j$. We add all possible edges between $V_{low}[i, j]$ and $A^{u_1}_{high}[u_1, u_2]$, and all possible edges between $V_{high}[i, j]$ and $A^{u_1}_{low}[u_1, u_2]$. In this way, $V_{low}[i, j]$ is connected to low order connector gadgets of vertex selection gadgets

and to high order connector gadgets of edge selection gadgets, and $V_{high}[i,j]$ is connected in the opposite way.

CNC instance: The graph H of our CNC instance is thus composed of $4\binom{\ell}{2}$ validation cliques which have $4\ell^4$ vertices each, n vertex selection gadgets each of size $(\ell-1)(2n^5+2n+2\ell^2)+n^2+1$, and m edge selection gadgets which have $2(2n^5+2n+2\ell^2)+n^2+1$ vertices each. We finish the description of our reduction by setting k, the size of the required critical node cut, to

$$k := \left(2(\ell-1)n + 4m - 8\binom{\ell}{2}\right)\cdot \ell^2 + \ell + \binom{\ell}{2},$$

and setting x, the bound on the number of connected pairs, to

$$x := \left(n+m-\ell-\binom{\ell}{2}\right)(n^2+1)n^2 + 4\binom{\ell}{2}(2n^5+2n+4\ell^4+2\ell^2)(2n^5+2n+4\ell^4+2\ell^2-1).$$

Lemma 1. *The graph H has treewidth at most $16\binom{\ell}{2}\ell^4+\ell^2+1$.*

Proof. In our proof we use two easy and well known facts about treewidth: The treewidth of graph is the maximum treewidth of all its components, and adding α vertices to a graph of treewidth at most β results in a graph of treewidth at most $\alpha+\beta$. Using these two facts we get that a connector gadget has treewidth at most ℓ^2, since we add ℓ^2 vertices to a graph of treewidth 0 (the independent set of vertices). From this we conclude that each selection gadget has treewidth at most ℓ^2+1, since we add a single vertex to a graph whose connected components have treewidth bounded by ℓ^2. Therefore, since H itself is constructed by adding $16\binom{\ell}{2}\ell^4$ validation vertices to a graph whose connected components have treewidth at most ℓ^2+1, the lemma follows. □

From a multicolored clique to a critical node cut: Suppose (G,c,ℓ) has a solution, *i.e.*, a multicolored clique S of size ℓ. Then one can verify that the cut $C \subseteq V(H)$ defined by

$$C := S \cup \{\{u_1,u_2\} : u_1 \neq u_2 \in S\} \cup \left\{v : v \in A^c_o[u], u \notin S\right\} \cup \left\{v : v \in A^u_o[u_1,u_2], u_1 \neq u_2 \notin S\right\}$$

is of size k, and $H-C$ contains exactly two types of connected components: $n+m-\ell-\binom{\ell}{2}$ components which include a single vertex from $V(G)\cup E(G)$ along with n^2 dummy vertices, and $4\binom{\ell}{2}$ components which have $2n^5+2n+4\ell^4+2\ell^2$ vertices each. Thus, $H-C$ has exactly x connected pairs, and C is indeed a solution to (H,k,x).

From a critical node cut to a multicolored clique: To complete the proof of Theorem 4, we show that if (H,k,x) has a solution, *i.e.*, a cut C of size k where $H-C$ has at most X connected pairs, then G has a multicolored clique of size ℓ. We do this, using a few lemmas that restrict the structure of solutions to our CNC instance. The first one of these, Lemma 2 below, shows that we can restrict our attention to cuts which include only core vertices of connector gadgets and vertices of $V(G)\cup E(G)$.

Lemma 2. *If there is a solution to (H,k,x), then there is a solution C to this instance which includes no guard vertices, no dummy vertices, and no validation vertices of H.*

Proof. Let C be a solution to (H,k,x). If C includes any dummy vertex v of H, then since v has degree one in H, we replace v with the neighbor of v (which is a non-dummy vertex) or if C contains the neighbor of v, we remove v from C. In both cases, the number of connected pairs in $H-C$ is not decreased. Similarly, if C includes guard vertices, these can be safely replaced with core vertices.

Next, we show that C also contains no vertices in validation cliques. First, we show that C cannot contain any validation clique completely. To this end, note that a core of a connector gadget which is not completely

included in C contributes more than n^{10} connected pairs in $H-C$. This can be seen by counting the number of connected pairs between a single core vertex and all its guard neighbors. Thus, since $(16\binom{\ell}{2}+1)n^{10} > x$ assuming a sufficiently large n, the cut C must include all but at most $16\binom{\ell}{2}$ cores of connector gadgets in H. But as each validation clique is of size $4\ell^4 > 8\binom{\ell}{2}\ell^2 + \ell + \binom{\ell}{2}$ (for sufficiently large ℓ), we have $k - 4\ell^4 < (2(\ell-1)n + 4m - 16\binom{\ell}{2})\ell^2$, which means that if C includes a validation clique it does not include enough cores. Thus, C cannot completely contain any validation clique.

Finally, consider the case that C contains a proper subset of vertices of some validation clique $V_o[i,j]$ in H. Observe first, that C completely contains all except at most two neighboring cores of $V_o[i,j]$: otherwise, the number of connected pairs is larger than $n^{15} > x$ for sufficiently large n. By the number of cores in H and by the value of k it also holds that at least $8 \cdot \binom{\ell}{2} - 1$ cores are not completely contained in C. Thus, there is no validation clique that is completely isolated in $H - C$. Hence, any vertex v in a validation clique that is not completely contained in C can be safely replaced by a core vertex that is adjacent to v as neither vertex is a cut vertex in $H - C \setminus \{v\}$. Altogether, vertices in validation cliques can be safely removed and the lemma follows. □

Assume that (H,k,x) has a solution, and fix a solution C as in Lemma 2. By the definition of k, we know that the cut C cannot include all connector gadgets. A connection gadget in $H-C$ induces a large number of connected pairs, at least n^{10}, due to the guard vertices of the gadget. Let us therefore call a connected component in $H-C$ *huge* if it contains at least n^{10} connected pairs. The next lemma shows that these huge components in $H-C$ has a very specific structure.

Lemma 3. *If C is a solution to (H,k,x) as in Lemma 2, then there are exactly $4\binom{\ell}{2}$ huge components in $H-C$. Furthermore, there exists a set of ℓ vertices $C_V \subseteq V(G)$, and a set of $\binom{\ell}{2}$ edges $C_E \subseteq E(G)$, $C_V \cup C_E \subseteq C$, where for each huge component Q in $H-C$ there is an $o \in \{low, high\}$, an $\bar{o} \in \{low, high\} \setminus \{o\}$, and an ordered pair (i,j), $i \neq j \in \{1,\ldots,\ell\}$, such that Q is composed of:*

- *A validation clique $V_o[i,j]$.*
- *A connector gadget $A^j_o[u] \cup B^j_o[u]$ of some u-selection gadget with $u \in C_V$ and $c(u) = i$.*
- *A connector gadget $A^{u_1}_{\bar{o}}[u_1,u_2] \cup B^{u_1}_{\bar{o}}[u_1,u_2]$ of some $\{u_1,u_2\}$-selection gadget with $\{u_1,u_2\} \in C_E$, $c(u_1) = i$, and $c(u_2) = j$.*

Proof. Let us call a maximal non-empty subset of a core in $H-C$ a *partial core*, and let $A_1,\ldots,A_t$ denote all partial cores in $H-C$. Note that since each core is of size $\ell^2 > \ell + \binom{\ell}{2}$, the cut C can include at most $(2(\ell-1)n + 4m - 8\binom{\ell}{2})$ complete cores by definition of k, and so $t \geq 8\binom{\ell}{2}$. By Lemma 2, the graph $H-C$ contains all $4\binom{\ell}{2}$ validation cliques in $H-C$. Let $Q_1,\ldots,Q_s$ denote the components in $H-C$ that contain at least one validation clique, and write $q_i = |Q_i| - 1$ for each i, $1 \leq i \leq s$. Then $s \leq 4\binom{\ell}{2}$, and the total number of connected pairs in all the Q_i's is lower bounded by $\sum_{i=1}^{s} q_i^2$. Now note that each partial core A_j belongs to some Q_i and contributes at least n^5+1 vertices to its size (accounting for a single vertex of A_j and all its guard neighbors). It can now be seen that since $\sum_{i=1}^{s} q_i^2$ is concave and symmetric, the only way for this sum to be less than $(16\binom{\ell}{2}+1)n^5 > x$ is if $t = 8\binom{\ell}{2}$, $s = 4\binom{\ell}{2}$, and each Q_i includes exactly two A_j's.

Consider a component Q that includes the validation clique $V_o[i,j]$ for some $o \in \{low, high\}$ and $i \neq j \in \{1,\ldots,\ell\}$, and let $\bar{o}$ be the element in $\{high, low\} \setminus \{o\}$. By construction of H, there are only two possibilities for partial cores that can belong to Q: a subset $A^i_o[u]$ for some vertex $u \in V(H)$ with $v(u) = i$, or a subset of $A^u_{\bar{o}}[u_1,u_2]$ for some edge $\{u_1,u_2\} \in E(G)$ with $c(u_1) = i$ and $c(u_2) = j$. Now, since $x = 16\binom{\ell}{2}n^{10} + O(n^6)$ and there are already $16\binom{\ell}{2}n^{10}$ connected pairs in $H-C$ which are caused by the partial cores, there cannot be a vertex from $V(G) \cup E(G)$ in Q as this will cause at least n^7 connected pairs between dummy and guard vertices. Thus, C must include at least one vertex $u \in V(H)$ with $c(u) = i$ and at least one vertex $\{u_1,u_2\} \in E(G)$ with $c(u_1) = i$ and $c(u_2) = j$.

Applying the argument above to each $o \in \{low, high\}$ and $i \neq j \in \{1,\ldots,\ell\}$, we get that C includes at least ℓ vertices of $V(G)$ and at least $\binom{\ell}{2}$ vertices from $E(G)$. Since $k - \ell - \binom{\ell}{2} = 2(\ell-1)n + 4m - 8\binom{\ell}{2}\ell^2$, and we already know that C has at least $2(\ell-1)n + 4m - 8\binom{\ell}{2}\ell^2$ core vertices, we get that there are exactly ℓ

vertices $C_V \subseteq C \cap V(H)$ and exactly $\binom{\ell}{2}$ vertices $C_E \subseteq C \cap E(H)$. Moreover, each partial core in $H - C$ is in fact a complete core, and each huge component is composed of a validation clique and two cores as specified in the lemma. □

Lemma 4. *If C is a solution to (H, k, x) as in Lemma 2, then there are exactly $n + m - \ell - \binom{\ell}{2}$ components in $H - C$ which are not huge nor trivial, each of which contains exactly $n^2 + 1$ vertices.*

Proof. According to Lemma 3, we know that each huge component in $H - C$ contains no vertices from $V(G) \cup E(G)$. Furthermore, C contains exactly ℓ vertices C_V from $V(G)$, and $\binom{\ell}{2}$ vertices C_E from $E(G)$. Thus, all vertices from $(V(G) \cup V(E)) \setminus (C_V \cup C_E)$ are present in $H - C$, and along with their dummy neighbors (which are not present in C according to Lemma 2) they each form a component in $H - C$ of size $n^2 + 1$. The lemma follows by observing that there are exactly $n + m - \ell - \binom{\ell}{2}$ such components, and no other non-huge non-trivial components in $H - C$ by construction. □

Lemma 5. *The set C_V specified in Lemma 3 induces a multicolored clique in G.*

Proof. First observe that C_V includes ℓ vertices of ℓ different colors. Thus, to prove the lemma above it suffices to show that C_V induces a clique in G. According to Lemma 4 above, the total number of connected pairs in components which are not huge in $H - C$ is exactly

$$\left(n + m - \ell - \binom{\ell}{2}\right)(n^2 + 1)n^2,$$

which means that the total number of connected pairs in huge components of $H - C$ is at most

$$4\binom{\ell}{2}(2n^5 + 2n + 4\ell^4 + 2\ell^2)(2n^5 + 2n + 4\ell^4 + 2\ell^2 - 1).$$

Note that due to Lemma 3, we know that there are $4\binom{\ell}{2}$ huge components in $H - C$, and that the total number of vertices in all of these huge components is $4\binom{\ell}{2}(2n^5 + 2n + 4\ell^4 + 2\ell^2)$.

Thus, the only way for the number of connected pairs in all huge components to not exceed the above bound is if all huge components have equal size, *i.e.*, exactly $(2n^5 + 2n + 4\ell^4 + 2\ell^2)$ vertices each. But this can happen only if we have $u = u'$ in the pair of connector guards $B^i_o[u]$ and $B^{u'}_o[u_1, u_2]$, in each huge component of $H - C$, as this is the only way for the guard vertices to sum up to $2n^5 + 2n$. This implies that $C_V = \{u : u \in \{u_1, u_2\} \in C_E\}$, which completes the proof of the lemma. □

Altogether the lemmas show that the construction is a reduction from Multicolored Clique parameterized by the number ℓ of clique vertices to Critical Node Cut instances with treewidth $O(\ell^6)$. This implies Theorem 4 and, using the results of Chor et al. [8], the following more explicit running time bound.

Corollary 2. *Assuming the exponential-time hypothesis,* Critical Node Cut *cannot be solved in $n^{o(\sqrt[6]{w})}$ time where w denotes the treewidth the input graph.*

4 Parameters $w + x$ and y

If we combine the treewidth parameter w with the parameter for the number of connected pairs x, then we obtain fixed-parameter tractability. The algorithm uses nice tree decompositions.

A *nice tree decomposition* [3] of a graph G is a pair $\langle \mathcal{X}, \mathcal{T} \rangle$, where each element $X \in \mathcal{X}$ (called a *bag*) is a subset of $V(G)$, and $\mathcal{T}$ is a rooted tree over $\mathcal{X}$. The pair $\langle \mathcal{X}, \mathcal{T} \rangle$ is required to satisfy the following conditions:

1. $\bigcup_{X \in \mathcal{X}} X = V(G)$.
2. For every edge $\{u, v\} \in E(G)$, there is an $X \in \mathcal{X}$ with $\{u, v\} \subseteq X$.
3. For all $X, Y, Z \in \mathcal{X}$, if Y lies on the path between X and Z in $\mathcal{T}$, then $X \cap Z \subseteq Y$.
4. There are 4 types of bags:

(a) A *leaf bag* X which has no children in $\mathcal{T}$ and contains a single vertex $v \in V(G)$.
(b) An *introduce bag* X which has a single child Y in $\mathcal{T}$ with $X = Y \cup \{v\}$ for some vertex $v \notin Y$.
(c) A *forget bag* X which has a single child Y in $\mathcal{T}$ with $X = Y \setminus \{v\}$ for some vertex $v \in Y$.
(d) A *join bag* X which has two children Y and Z in $\mathcal{T}$ with $X = Y = Z$.

Note that Conditions $1-3$ define a tree decomposition and their combination with Condition 4 defines a nice tree decomposition. The width of a tree decomposition is the number of elements in the largest bag minus 1. The treewidth w of G is the minimum width of all the possible tree decompositions of G. For a given graph G with treewidth w, one can obtain it's nice tree decomposition in $f(w) \cdot n^{O(1)}$ with $O(wn)$ bags [3,4]. Thus, in proving the main result of this section, stated in the theorem below, we can assume that we are given as input a nice tree decomposition $\langle \mathcal{X}, \mathcal{T} \rangle$ of width w (and $O(wn)$ bags) of our input graph G.

Theorem 5. *The* Critical Node Cut *problem is FPT with respect to* $w + x$.

Let X be a bag from our nice tree decomposition, and let G_X denote the subgraph of G induced by the bag X and all of its descendants in $\mathcal{T}$. We build a table for X. Each entry in this table is denoted by $T_X[k', x', X_0, X_1, \ldots, X_t, n_1, \ldots, n_t]$, where $k' \leq k$, $x' \leq x$, and $n_i \leq x$ and $X_i \subseteq X$ for all i. The entry can either equal 0 or 1. It equals 1 iff there exist k' vertices C in G_X, $X_0 \subseteq C$, such that $G_X - C$ has at most x' connected pairs and is separated into components $R_1, R_2, \ldots, R_t$, with $X \cap R_i = X_i$ and $|R_i| = n_i$ for each i, $1 \leq i \leq t$. If there is no such solution then the entry equals 0. Thus, an entry with value 1 corresponds to a partial solution that splits the bag X in a very particular way. Our algorithm calculates the tables of each node in the decomposition in a bottom-up fashion. Clearly, if each entry is computed correctly then one can infer whether there exists a solution to the CNC instance by examining the table at the root. Note that for each bag the size of the table is $O(nx(w+x)^w)$, thus if we show that calculating an entry can be done in FPT time, then we prove that our algorithm altogether runs in FPT time.

We will next show how to calculate T_X for each possible type of bag X. If X is a leaf bag, then X is composed of one vertex only, and therefore the computation in this case is trivial. If X is a forget bag with a child Y such that $Y = X \cup \{v\}$ for $v \notin X$, then the entry $T_X[k', x', X_0, X_1, \ldots, X_i, \ldots, X_t, n_1, \ldots, n_t]$ will equal 1 if and only if there exists in the table of Y an entry $T_Y[k', x', X_0, X_1, \ldots, X_i \cup \{v\}, \ldots, X_t, n_1, \ldots, n_t]$ that equals 1. This is correct because the only difference between the entries is that $v \notin X$, therefore it is excluded from it's partition, yet it is still counted by the value n_i as a member of the corresponding component. To complete the proof of Theorem 5, we show in the next two lemmas that we can efficiently calculate each entry in T_X also if X is an introduce or a join bag.

Lemma 6. *If X is an introduce bag with child Y, then, given the table T_Y, an entry in T_X can be calculated in $|T_y|^{O(1)}$ time.*

Proof. Let v be the single vertex in $X \setminus Y$, and consider an arbitrary entry $T_X[k', x', X_0, X_1, \ldots, X_t, n_1, \ldots, n_t]$ in the table of X. There are three possible cases:

1. $v \in X_0$. The entry in T_X will equal 1 iff in T_Y we have
$$T_Y[k' - 1, x', X_0 \setminus \{v\}, X_1, \ldots, X_t, n_1, \ldots, n_t] = 1.$$
2. $v \notin X_0$ and v is not adjacent to any vertex in G_Y. The only type of entry T_X that might have value 1 is an entry where $X_i = \{v\}$ for some $i \geq 1$, and $n_i = 1$. If this is the case then the current entry in T_X will be 1 iff we have
$$T_Y[k', x', X_0, \ldots, X_{i-1}, X_{i+1}, \ldots, X_t, n_1, \ldots, n_{i-1}, n_{i+1}, \ldots, n_t] = 1.$$
3. $v \notin X_0$ and v is adjacent to vertices of Y. By the properties of a tree decomposition, in G_X the vertex v can only have neighbors from Y. Let $y_1, \ldots, y_a \in Y$ denote these neighbors. Then the only entries in T_X that might have value 1 are entries with a subset $X_i \subseteq X$ that includes x and all his neighbors. If the current entry in T_X is as such, then it will have value 1 iff
$$T_Y[k', x'', X_0, X_1, \ldots, X_{i-1}, Y_1, \ldots, Y_b, X_{i+1}, \ldots, X_t, n_1, \ldots, n_{i-1}, m_1, \ldots, m_b, n_{i+1}, \ldots, n_t,] = 1,$$

for an entry in T_Y where $Y_1, \ldots, Y_b$ are precisely the subsets of Y that include neighbors of v, $X_i = \{v\} \cup \bigcup_j Y_j$, $x'' = x' - 2\sum_j m_j - \sum_j m_j \sum_{k \neq j} m_k$, and $n_i = 1 + \sum_j m_j$.

The correctness of the first two cases is easy to see. The reason that Case 3 is correct is that if the entry from Y exists then adding v will connect the components $R_1, \ldots, R_a$ in G_Y corresponding to $Y_1, , \ldots, Y_a$ above into one component X_i. The size of the new component will be the sum of the sizes of the previous components plus 1 (because of v). The number of connected pairs added in the process is the number of connected pairs between v and all vertices in $\bigcup_j R_j$ $(2\sum_{j=1}^{b} m_j)$, and the number of connections between vertices in different components $(\sum_{j=1}^{b} m_j \sum_{k \neq j} m_k)$. Conversely, any given entry of value 1 in T_X must correspond to an entry from T_Y as stated above. Since the computation of the entry in T_X is clearly polynomial in the size of T_Y, the lemma follows. □

Lemma 7. *If X is a join node with children Y and Z, then, given the tables T_Y and T_Z, an entry in T_X can be calculated in $(|T_y| + |T_z|)^{O(1)}$ time.*

Proof. Recall that by definition we have $X = Y = Z$. Let $T_X[k', x', X_0, X_1, \ldots, X_t, n_1, \ldots, n_t]$ be an arbitrary entry in the table of X. This entry equals 1 iff there exist in T_Y and T_Z the entries $T_Y[k'_Y, x'_Y, Y_0, Y_1, \ldots, Y_p, n'_1, \ldots, n'_p]$ and $T_Z[k'_Z, x'_Z, Z_0, Z_1, \ldots, Z_q, n''_1, \ldots, n''_q]$ that equal 1 and satisfy the following conditions:

1. $X_0 = Y_0 \cup Z_0$ and also $k = k'_Y + k'_Z - |Y_0 \cap Z_0|$.
2. We define a relation $\approx$ on the vertices of X given by $u \approx v$ iff $\{u, v\} \in Y_i$ or $\{u, v\} \in Z_j$ for $i \in \{1, \ldots, p\}$ and $j \in \{1, \ldots, q\}$. The partition $X_1, \ldots, X_t$ is required to be defined by the equivalence classes of the transitive closure of this relation.
3. The size n_i of each component corresponding to X_i match up according to the differences between the new partition and the old ones using the exclusion-inclusion principle. That is, if $X_i = Y_{j_1} \cup \ldots \cup Y_{j_\alpha} \cup Z_{k_1} \cup \ldots \cup Z_{k_\beta}$, then let $A_s = Y_{j_s}$ for $s \in \{1, \ldots, \alpha\}$ and $A_s = Z_{k_{s-\alpha}}$ for $s \in \{\alpha + 1, \ldots, \alpha + \beta\}$, and we require that
$$n_i = \sum_{\ell=1}^{\alpha} n'_{j_\ell} + \sum_{\ell=1}^{\beta} n''_{k_\ell} + \sum_{\substack{L \subseteq \{1,\ldots,\alpha+\beta\}, \\ |L| \geq 2}} (-1)^{|L|-1} \left| \bigcap_{\ell \in L} A_\ell \right|.$$
4. The bound on the number of connected pairs x' adds up in the correct way. That is,
$$x' = x'_Y + x'_Z + \sum_{i=1}^{t} n_i(n_i - 1) - \sum_{j=1}^{p} n'_j(n'_j - 1) - \sum_{k=1}^{q} n''_k(n''_k - 1).$$

Note that for the given entry in T_X, we can verify whether there exist appropriate entries in T_Y and T_Z that satisfy the four requirement above in polynomial time with respect to the sizes of this tables. Furthermore, it can be readily verified that if $R_1^Y \ldots, R_p^Y$ are connected components in G_Y corresponding to a partition $\{Y_0, Y_1, \ldots, Y_p\}$ of Y, and $R_1^Z \ldots, R_q^Z$ are connected components in G_Z corresponding to a partition $\{Z_0, Z_1, \ldots, Z_q\}$ of Z, then in G_X we will have t connected components $R_1^X \ldots, R_t^X$ that correspond to the partition $\{X_0, X_1, \ldots, X_t\}$ defined above. This follows directly from the fact that connectivity is an equivalence relation, and the fact that a component R_j^Y can intersect a component R_k^Z only at vertices of X. Thus, since all other requirements are direct corollaries of this fact, the existence of two such entries in T_Y and T_Z imply that the current entry in T_X equals 1. The converse implication, that is, the fact that if the current entry of T_X equals 1 there must exist two entries in T_Y and T_Z which equal 1 and satisfy the above requirements, follows along the same lines. □

5 Parameter y

Finally, we consider the CNC problem parameterized by y. We will show that the problem is FPT under this parameterization but has no polynomial kernel even for the aggregate parameterization of $k + y + w$.

Theorem 6. *The* Critical Node Cut *problem is FPT with respect to y.*

Proof. Let (G, k, y) be an instance of CNC. Observe that if one of the components was larger than y, then removing one vertex from this components already causes the removal of at least y connected pairs. Moreover, if k was larger than y (in fact, $y/2$) then removing any k arbitrary non-isolated vertices has the same effect. Thus, the interesting case occurs when $k < y$ and each component of G has size at most y, and we assume henceforth throughout the proof that this is in fact the case.

Our algorithm proceeds as follows, herein let $G_1, \ldots, G_t$ denote the connected components of G:

1. For each component G_i of G and each k', $1 \leq k' \leq k$, compute by brute-force the maximum number of connected pairs in G_i that can be removed by deleting exactly k' vertices in G_i. Let $T[i, k']$ denote this number.
2. For increasing i, compute the maximum number of connected pairs that can be removed by deleting exactly k' vertices in the components $G_1, \ldots, G_i$. Let $Q[i, k']$ denote this number. For $i = 1$, we have $Q[1, k'] = T[1, k']$. For $i > 1$, we have
$$Q[i, k'] = \min_{k'' \leq k'} Q[i-1, k''] + T[i, k' - k''].$$
3. If $Q[t, k] < y$ return NO; otherwise, return YES.

Correctness of the algorithm is rather obvious: optimal solutions for different components can be combined since the connected pairs are only contained within each component. The running time is $O(2^y \cdot y^2 n)$: Each component has at most y vertices, thus there are $O(2^y)$ possibilities to consider in the brute-force step. For each possibility, computing the size of the remaining connected components can be done in $O(y^2)$ time. The dynamic programming in the second step of the algorithm is then performed for $t \leq n$ different values of i. For each value of i, $k^2 \leq y^2$ possible combinations of k' and k'' are considered. □

Using the composition technique developed in [5], we now show that parameter y does not seem to be useful when considering polynomial kernelization for CNC. In this technique one attempts to exclude polynomial kernels by showing that they admit a form of composition algorithm. Since there are many types of composition algorithms with small differences, we give here a definition which is tailor suited for our purposes.

Definition 1 ([5,9]). *A* composition algorithm *for* CNC *parameterized by $k + y + w$ is a polynomial time algorithm that receives as input a sequence of instances $(G_1, k, y), \ldots, (G_t, k, y)$ for* CNC*, where each graph G_i has n vertices, and outputs an instance (G, k', y') of* CNC *such that:*

- *(G, k', y') is a yes-instance iff (G_i, k, y) is a yes-instance for some i, and*
- *$k' + y' + w = n^{O(1)}$, where w is the treewidth of G.*

Using the results in [5,9,14], it immediately follows that a composition algorithm for CNC parameterized by $k + y + w$ along with a polynomial kernel for the problem causes the polynomial hierarchy to collapse to its third level, a consequence that is considered highly unlikely according to our current state of knowledge. We use this in order to prove the theorem below:

Theorem 7. *The* Critical Node Cut *problem parameterized by $k+y+w$ has no polynomial kernel unless the polynomial hierarchy collapses.*

Proof. We present a composition algorithm for the problem. Let $(G_1, k, y), ..., (G_t, k, y)$ be instances of CNC given as input to our composition algorithm, and let n denote the number of vertices in each instance. We construct the output instance (G, k', y') in the following way: We start the construction of G by taking the disjoint union of the input graphs G_i. Next, to each G_i we add a clique K_i of size $k + 1$ and add all possible edges between vertices of G_i and vertices of K_i. Finally, we set $y' = y + (k + 1)n$ and $k' = 2k + 1$.

Clearly, if there exists a solution C of to one of the input instances (G_i, k, y), then $C \cup K_i$ is a solution of to (G, k', y'), since C is of size $2k+1$ and it removes $y' = y+2(k+1)n$ connected pairs in G (accounting also

for the connected pairs involving vertices of K_i). In the other direction, if there exists a solution C to the output instance then we can assume it removes vertices from a single graph $G_i \cup K_i$. This is because each instance remains connected so long as the clique attached to it has not been deleted, and the bound k' allows only for the removal of a single clique K_i. This implies that (G_i, k, y) is a yes instance. Obviously, $y = O(n^2)$, $k = O(n)$, and $w = O(n + k) = O(n^2)$, where w is the treewidth of G. Thus, $k + y + w = n^{O(1)}$. □

6 Discussion

We considered a natural graph cut problem called Critical Node Cut (CNC) under the framework of parameterized complexity. The only parameterization left open in our analysis is the parameter $w + k$, and so the first natural question left open in the paper is whether CNC is fixed-parameter tractable under this parameterization (we know it is unlikely that it admits a polynomial kernel). Other natural parameters could also be considered. For example, it would be interesting to see how parameters *maximum degree* and *pathwidth* affect the parameterized complexity of CNC. Finally, one can consider the edge variant of the problem (where one is required to delete edges instead of vertices) and the directed variant of the problem. Many of our results do not hold for these two variants.

References

1. Bernardetta Addis, Marco Di Summa, and Andrea Grosso. Removing critical nodes from a graph: complexity results and polynomial algorithms for the case of bounded treewidth. *Optimization online (www.optimization-online.org)*, 2011.
2. Ashwin Arulselvan, Clayton W Commander, Lily Elefteriadou, and Panos M Pardalos. Detecting critical nodes in sparse graphs. *Computers & Operations Research*, 36(7):2193–2200, 2009.
3. Hans L Bodlaender. A tourist guide through treewidth. *Acta Cybernetica*, 11(1-2):1, 1994.
4. Hans L. Bodlaender. A linear time algorithm for finding tree-decompositions of small treewidth. *SIAM Journal on Computing*, 25:1305–1317, 1996.
5. Hans L. Bodlaender, Rodney G. Downey, Michael R. Fellows, and Danny Hermelin. On problems without polynomial kernels. *Journal of Computer and System Sciences*, 75(8):423–434, 2009.
6. Vladimir Boginski and Clayton W Commander. Identifying critical nodes in protein–protein interaction networks. *Clustering challenges in biological networks*, pages 153–167, 2009.
7. Jonathan F Buss and Judy Goldsmith. Nondeterminism within P. *SIAM Journal on Computing*, 22(3):560–572, 1993.
8. Jianer Chen, Benny Chor, Mike Fellows, Xiuzhen Huang, David W. Juedes, Iyad A. Kanj, and Ge Xia. Tight lower bounds for certain parameterized np-hard problems. *Inf. Comput.*, 201(2):216–231, 2005.
9. Holger Dell and Dieter van Melkebeek. Satisfiability allows no nontrivial sparsification unless the polynomial-time hierarchy collapses. In *Proc. 42nd STOC*, pages 251–260, 2010.
10. Marco Di Summa, Andrea Grosso, and Marco Locatelli. Complexity of the critical node problem over trees. *Computers & Operations Research*, 38(12):1766–1774, 2011.
11. Rodney G. Downey and Michael R. Fellows. *Parameterized Complexity.* Springer-Verlag, 1999.
12. Andrew Drucker. New limits to classical and quantum instance compression. In *Proc. 53rd FOCS*, 2012.
13. Michael R. Fellows, Danny Hermelin, Frances A. Rosamond, and Stéphane Vialette. On the parameterized complexity of multiple-interval graph problems. *Theoretical Computer Science*, 410(1):53–61, 2009.
14. Lance Fortnow and Raul Santhanam. Infeasibility of instance compression and succinct PCPs for NP. In *Proc. 40th STOC*, pages 133–142, 2008.
15. M.R. Garey and D.S. Johnson. *Computers and intractability; a guide to the theory of NP-completeness.* W.H. Freeman, 1979.
16. Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *Journal of Computer and Systems Sciences*, 63(4):512–530, 2001.
17. Karen E Joyce, Paul J Laurienti, Jonathan H Burdette, and Satoru Hayasaka. A new measure of centrality for brain networks. *PLoS One*, 5(8):e12200, 2010.
18. Vito Latora and Massimo Marchiori. How the science of complex networks can help developing strategies against terrorism. *Chaos, solitons & fractals*, 20(1):69–75, 2004.

19. Cullen M Taniguchi, Brice Emanuelli, and C Ronald Kahn. Critical nodes in signalling pathways: insights into insulin action. *Nature reviews Molecular cell biology*, 7(2):85–96, 2006.
20. Mario 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.
21. Mario Ventresca and Dionne Aleman. A derandomized approximation algorithm for the critical node detection problem. *Computers & Operations Research*, 43:261–270, 2014.