Faster exact algorithms for some terminal set problems

Chitnis, Rajesh; Fomin, Fedor V.; Lokshtanov, Daniel; Misra, Pranabendu; Ramanujan, M. S.; Saurabh, Saket

DOI:
10.1016/j.jcss.2017.04.003

License:
Other (please provide link to licence statement

Document Version
Peer reviewed version

Citation for published version (Harvard):
Chitnis, R, Fomin, FV, Lokshtanov, D, Misra, P, Ramanujan, MS & Saurabh, S 2017, 'Faster exact algorithms for some terminal set problems', *Journal of Computer and System Sciences*, vol. 88, no. September, pp. 195-207. https://doi.org/10.1016/j.jcss.2017.04.003

Faster Exact Algorithms for Some Terminal Set Problems *

Rajesh Chitnis[†1], Fedor V. Fomin[2], Daniel Lokshtanov[2], Pranabendu Misra[‡3], M.S. Ramanujan[§2], and Saket Saurabh[2,3]

[1]Faculty of Mathematics and Computer Science, Weizmann Institute of Science, Israel. Email: {rajesh.chitnis}@weizmann.ac.il
[2]Department of Informatics, University of Bergen, Norway. Email: {fomin,daniello}@ii.uib.no, ramanujan@ac.tuwien.ac.at
[3]The Institute of Mathematical Sciences, HBNI, Chennai, India. Email: {pranabendu,saket}@imsc.res.in

Abstract

Many problems on graphs can be expressed in the following language: given a graph $G = (V, E)$ and a terminal set $T \subseteq V$, find a minimum size set $S \subseteq V$ which intersects all "structures" (such as cycles or paths) passing through the vertices in T. We refer to this class of problems as terminal set problems. In this paper, we introduce a general method to obtain faster exact exponential time algorithms for several terminal set problems. In the process, we break the $\mathcal{O}^*(2^n)$ barrier for the classic Node Multiway Cut, Directed Unrestricted Node Multiway Cut and Directed Subset Feedback Vertex Set problems.

1 Introduction

The goal of the design of moderately exponential time algorithms for NP-complete problems is to establish algorithms for which the worst-case running time is provably faster than the one of enumerating all prospective solutions, or loosely speaking, algorithms better than brute-force enumeration. For example, for NP-complete problems on graphs on n vertices and m edges whose solutions are either subsets of vertices or edges, the brute-force or trivial algorithms basically enumerate all subsets of vertices or edges. This mostly leads to algorithms of time complexity $\mathcal{O}^*(2^n)$ or $\mathcal{O}^*(2^m)$, based on whether we are enumerating vertices or edges[1]. Thus the goal of exact algorithms for graph problems is to improve upon the algorithms running in time $\mathcal{O}^*(2^n)$ or $\mathcal{O}^*(2^m)$. See the book [1] for an introduction to exact exponential algorithms.

One of the most well studied directions in exact algorithms is to delete vertices of the input graph such that the resulting graph satisfies some interesting properties. More precisely, a natural

*The research of Fedor V. Fomin leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013) / ERC Grant Agreement n. 267959. Daniel Lokshtanov is supported by the BeHard grant under the recruitment programme of the of Bergen Research Foundation. Saket Saurabh is supported by PARAPPROX, ERC starting grant no. 306992.

[†]Most of the work was done while the author was at the University of Maryland.

[‡]This author has moved to the Department of Informatics, University of Bergen.

[§]This author has moved to the Algorithms and Complexity Group, TU Wien.

[1]Throughout this paper we use the $\mathcal{O}^*$ notation which suppresses polynomial factors, i..e, $f(n) = \mathcal{O}^*(g(n))$ if $f(n) = \mathcal{O}(g(n) \cdot \text{poly}(n))$

optimization problem associated with a graph class $\mathcal{G}$ is the following: given a graph G, what is the minimum number of vertices to be deleted from G to obtain a graph in $\mathcal{G}$? For example, when $\mathcal{G}$ is the class of empty graphs, forests or bipartite graphs, the corresponding problems are Vertex Cover (VC), Feedback Vertex Set (FVS) and Odd Cycle Transversal (OCT), respectively. The best known algorithms for VC, FVS and OCT run in time $\mathcal{O}^*(1.2002^n)$ [2], $\mathcal{O}^*(1.7347^n)$ [3] and $\mathcal{O}^*(1.4661^n)$ [4, 5] respectively. The other problems in this class for which non-trivial exact algorithms are known include finding an induced r-regular subgraph [6], induced subgraph of bounded degeneracy [7] and induced subgraph of bounded treewidth [3].

In this paper we study another class of graph problems which we call *terminal set problems*. In these problems, the input is a graph $G = (V, E)$ and a terminal set $T \subseteq V$, and the goal is to find a minimum size set $S \subseteq V$ that intersects certain structures such as cycles or paths passing through the vertices in T. In this paper we introduce a general method to obtain faster exact exponential time algorithms for many terminal set problems. The general algorithmic technique is the following. Let the size of the terminal set T be k. We first observe that the size of the optimum solution is at most k (or a function of k). Let S be an optimum solution to the problem and let $X = S \setminus T$. We guess X and delete it from G. Since $S \setminus X \subseteq T$, we create an auxiliary graph on T and determine the rest of the solution using either a known polynomial time algorithm, or a fixed parameter tractable algorithm, or a non-trivial exact algorithm for the non-terminal version (when $T = V$) of the same problem. We now provide a list of problems for which we give improved or new algorithms using our method together with a short overview of previous work on each application.

Node Multiway Cut and Directed Unrestricted Node Multiway Cut: A fundamental min-max theorem about connectivity in graphs is Menger's Theorem, which states that the maximum number of vertex disjoint paths between two vertices s and t, is equal to the minimum number of vertices whose removal separates these two vertices. Indeed, a maximum set of vertex disjoint paths between s, t and a minimum size set of vertices separating these two vertices can be computed in polynomial time. A known generalization of this theorem, commonly known as Mader's T-path Theorem [8] states that, given a graph G and a subset T of vertices, there are either k vertex disjoint paths with only the end points in T (such paths are called *T-paths*), or there is a vertex set of size at most $2k$ which intersects every T-path. Although computing a maximum set of vertex disjoint T-paths can be done in polynomial time by using matching techniques, the decision version of the dual problem of finding a minimum set of vertices that intersects every T-path is NP-complete for $|T| > 2$. Formally, this problem is the following classical Node Multiway Cut problem.

Node Multiway Cut(NMC)
Input: An undirected graph $G = (V, E)$ and a set of terminals $T = \{t_1, t_2, \ldots, t_k\}$.
Question: Find a set $S \subseteq V \setminus T$ of minimum size such that $G \setminus S$ has no path between a t_i, t_j pair for any $i \neq j$.

This is a very well studied problem in terms of approximation, as well as parameterized algorithms [9, 10, 11, 12]. A variant of this problem where S is allowed to intersect the set T, is known as Unrestricted Node Multiway Cut (UNMC). The best known parameterized algorithm for Node Multiway Cut decides in time $\mathcal{O}^*(2^\ell)$ whether there is a solution of size at most ℓ or not. [13] designed an exact algorithm for UNMC running in time $\mathcal{O}^*(1.8638^n)$. We improve on this by designing an algorithm which runs in time $\mathcal{O}^*(1.4767^n)$ for UNMC. We also design an algorithm

with running time $\mathcal{O}^*(1.4767^n)$ for NMC: this is the first algorithm that improves upon the trivial $\mathcal{O}^*(2^n)$ algorithm for this problem.

Next we consider the directed variant of Node Multiway Cut, namely Directed Node Multiway Cut (DNMC) where the input is a directed graph and a set $T = \{t_1, \ldots, t_k\}$ of terminals and the objective is to find a set of vertices of minimum size which intersects every $t_i \to t_j$ path for every $t_i, t_j \in T$ with $i \neq j$. For the *unrestricted* version of this problem, namely Directed Unrestricted Node Multiway Cut (DUNMC), we design an exact algorithm with running time $\mathcal{O}^*(1.6181^n)$ which is the first algorithm improving upon the trivial $\mathcal{O}^*(2^n)$ algorithm for this problem. The problem of designing any algorithm breaking the trivial $\mathcal{O}^*(2^n)$ barrier for DNMC is still open.

Subset Feedback Vertex Set and Directed Subset Feedback Vertex Set: A nontrivial exact algorithm for Feedback Vertex Set (FVS) – finding a minimum sized vertex subset such that its removal results in an acyclic graph – remained elusive for several years. In a breakthrough paper Razgon [14] designed an exact algorithm for this problem running in time $\mathcal{O}^*(1.8899^n)$. Later, Fomin et al. [15] building upon the work in [14] designed an exact algorithm for FVS running in time $\mathcal{O}^*(1.7548^n)$. The current best known algorithm for this problem uses potential maximal clique machinery and runs in time $\mathcal{O}^*(1.7347^n)$ [3]. Razgon studied the directed version of FVS and obtained an exact algorithm running in time $\mathcal{O}^*(1.9977^n)$ [16]. This is the only known non-trivial exact algorithm for Directed Feedback Vertex Set (DFVS). A natural generalization of the Feedback Vertex Set problem is when we only want to hit all the cycles passing through a specified set of terminals. This leads to the following problem.

> **Subset Feedback Vertex Set(SFVS)**
> *Input*: An undirected graph $G = (V, E)$, a set of terminals $T \subseteq V$ of size k
> *Question*: Find a minimum set of vertices which hits every cycle passing through T

Fomin et al. [13] designed an algorithm for SFVS on general graphs which runs in time $\mathcal{O}^*(1.8638^n)$. It is important to note that their algorithm not only finds a minimum sized solution, but also enumerates all minimal solutions in the same time. Using our methodology we design an algorithm for SFVS which runs in time $\mathcal{O}^*(1.8932^n)$, and if we allow randomization then we can design an algorithm with an expected running time of $\mathcal{O}^*(1.8826^n)$. While our algorithms are currently not the fastest ones, as faster FPT or exact algorithm for FVS are devised, the running time of our algorithm will continue to improve. Since the running time of our algorithm for SFVS is quite close to the algorithm of [13], we expect that the further improvements in the algorithms for FVS will allow our algorithm to take the lead. Golovach et al. [17] initiated the study of exact algorithms for SFVS on special graph classes by giving an enumeration algorithm for SFVS on chordal graphs which runs in time $\mathcal{O}^*(1.6708^n)$. They left it as an open question whether there exist algorithms for SFVS on chordal graphs (even on split graphs) which are faster than $\mathcal{O}^*(1.6708^n)$. Though our algorithm using the described methodology for SFVS, in general, does not improve on the best known algorithm, it answers this question in the affirmative for SFVS on chordal graphs. In particular, we obtain an algorithm with running time $\mathcal{O}^*(1.6181^n)$. More generally, our algorithm for SFVS runs in $\mathcal{O}^*(1.6181^n)$ on any graph class $\mathcal{G}$ which is closed under vertex deletions and edge contractions, and where the weighted FVS problem can be solved in polynomial time. Finally, we also consider the directed variant of the SFVS problem, namely Directed Subset Feedback Vertex Set (DSFVS), and obtain an algorithm with running time $\mathcal{O}^*(1.9993^n)$. This is the first algorithm improving upon the trivial $\mathcal{O}^*(2^n)$ algorithm for this problem.

2 Preliminaries

Let C be a cycle in a graph G. A *chord* of C is an edge $e \notin C$ which connects two vertices of C. A graph G is called a *chordal graph* if every cycle on four or more vertices has a *chord*. A clique is a graph that has an edge between every pair of vertices, and we use K_n for $n \geq 2$ to denote a clique on n vertices.

Now we define the *contraction* of an edge or a subgraph in G. Let G be an undirected graph and let (u, v) be an edge in G. Let G' be the graph obtained from G in the following manner. We add a new vertex w. For every edge (u, z) where $z \neq v$ we add an edge (w, z), and for every edge (y, v) where $y \neq u$ we add an edge (y, w). Finally we delete the vertices u and v, and any parallel edges from the graph. We say that G' is obtained from G by *contracting* the edge (u, v). Let H be a subgraph of G. Consider the graph G' obtained from G by contracting every edge of H in an arbitrary order. We say that G' is obtained from G by contracting the subgraph H.

Now we define the *torso graph* of a subset of vertices in G. Let $G = (V, E)$ be an undirected graph and T and V' be subsets of V. We denote by $\texttt{torso}(T, V')$ the graph defined in the following manner. The vertex set of this graph is T and the edge set comprises of all pairs (t_i, t_j) such that there is a $t_i - t_j$ path in G whose internal vertices lie in $V' \setminus T$ or there is an edge $(t_i, t_j) \in E$.

We also define an analogous notion of a *torso graph* in directed graphs. Let $D = (V, A)$ be a directed graph and T and V' be subsets of V. We denote by $\texttt{torso}(T, V')$ the digraph defined in the following manner. The vertex set of this graph is T and the edge set comprises of all ordered pairs (t_i, t_j) such that there is a directed $t_i - t_j$ path in D whose internal vertices lie in $V' \setminus T$ or there is an edge $(t_i, t_j) \in A$.

Finally, we define a generalization of the *torso graphs*. Let $G = (V, E)$ be an undirected graph and $T_1, T_2, \ldots T_\ell$ and V' be subsets of V. We denote by $\ell\text{-}\texttt{torso}(T_1, \ldots, T_\ell, V')$ the graph defined as follows. It has vertex set $T = T_1 \cup T_2 \cup \cdots \cup T_\ell$. and the edge set comprises of all pairs (t_i, t_j) such that $t_i \in T_{i'}$ and $t_j \in T_{j'}$ for some $i' \neq j'$ and there is a $t_i - t_j$ path in G whose internal vertices lie in $V' \setminus T$.

3 Node Multiway Cut

In this section we design an exact algorithm for the Node Multiway Cut problem. We begin by giving an algorithm for unrestricted version of this problem.

3.1 Unrestricted Node Multiway Cut

The following observation follows from the fact that the set of terminals in an instance of Unrestricted Node Multiway Cut itself is a potential solution.

Observation 3.1. *Let (G, T) be an instance of* Unrestricted Node Multiway Cut *and S be an optimum solution to this instance . Then $|S| \leq |T|$.*

Now we design an algorithm for Unrestricted Node Multiway Cut using the FPT algorithm for Node Multiway Cut and Observation 3.1. This algorithm uses the FPT algorithm for multiway cut of [10]. We will use this algorithm for the instances where k is "small".

Lemma 3.2. *Let (G, T) be an instance of* Unrestricted Node Multiway Cut *where $|T| = k$. Then we can find an optimum solution to this instance in time $\mathcal{O}^*(2^k)$.*

Proof. We begin by performing the following transformation on the given instance. For each terminal $t_i \in T$, we add a new vertex t_i' and make it adjacent to t_i alone. Let T' be the set of these new

vertices and G' be the graph thus constructed. It is easy to see that the set of optimum solutions to the instance (G', T') of Node Multiway Cut and the set of optimum solutions to the instance (G, T) of Unrestricted Node Multiway Cut are in one to one correspondence. Therefore, we may utilise the algorithm of [10] which solves Node Multiway Cut in time $\mathcal{O}^*(2^\ell)$ where ℓ is the size of the optimum solution. By Observation 3.1, we have that $\ell \leq k$ and therefore, the statement of the lemma follows. □

Next, we design another algorithm for Unrestricted Node Multiway Cut which will be used for the instances where k is "large".

Lemma 3.3. *Let (G, T) be an instance of* Unrestricted Node Multiway Cut *where $G = (V, E)$ and let S be an optimum solution to this instance. Let $X = S \setminus T$ and $Y = S \cap T$. Then Y is a vertex cover of the graph $\texttt{torso}(T, V \setminus X)$. Conversely, if Y' is any vertex cover for the graph $\texttt{torso}(T, V \setminus X)$, then the set $X \cup Y'$ is a solution to this instance.*

Proof. We first show that Y is indeed a vertex cover of $G' = \texttt{torso}(T, V \setminus X)$. Let E' be the edge set of G'. Suppose that Y is not a vertex cover of G' and there is an edge $(t_i, t_j) \in E'$ which is not covered by Y. Observe that $(t_i, t_j) \notin E$, since this would contradict the assumption of S being a solution. Therefore, it must be the case that there is a path P between t_i and t_j in $G[V \setminus X]$ whose internal vertices are disjoint from T. Since this path is disjoint from both X and Y, it is also present in the graph $G \setminus S$, a contradiction. Hence, we conclude that Y is indeed a vertex cover of $\texttt{torso}(T, V \setminus X)$.

We now show that for any vertex cover Y' of G', the set $X \cup Y'$ is a solution to the instance (G, T) of Unrestricted Node Multiway Cut. Suppose to the contrary that there is a vertex cover Y' of $G' = \texttt{torso}(T, V \setminus X)$ such that the set $S' = X \cup Y'$ is not a solution to the instance (G, T). That is, there is a t_i-t_j path in $G \setminus S'$ for some $t_i, t_j \in T$. Observe that this implies the existence of a $t_{i'}$-$t_{j'}$ path P in G for some $t_{i'}, t_{j'} \in T \setminus Y'$ such that the internal vertices of P (if any) are disjoint from $T \cup S'$. Therefore the edge $(t_{i'}, t_{j'})$ is not covered by Y' in G', a contradiction. This completes the proof this lemma. □

Using the above lemma and the FPT algorithm for Vertex Cover of [18], we are able to show the following lemma.

Lemma 3.4. *There is an algorithm that given an instance $(G = (V, E), T)$ of* Unrestricted Node Multiway Cut *returns an optimum solution in time $\mathcal{O}^*\left(1.7851^n\left(\frac{1.2738}{1.7851}\right)^k\right)$, where $k = |T|$ and $n = |V|$.*

Proof. The description of the algorithm is as follows. For every $X \subseteq (V \setminus T)$ such that $|X| \leq k$, we construct the graph $G_X = \texttt{torso}(T, V \setminus X)$ and compute a minimum vertex cover Y_X for G_X. We compute the minimum vertex cover by using the FPT algorithm of [18], which runs in time $\mathcal{O}^*(1.2738^\ell)$ where ℓ is the size of an optimum vertex cover. Finally, we return the set $X \cup Y_X$ which is a smallest solution over all choices of X. The correctness of this algorithm follows from Lemma 3.3.

In order to bound the running time of this algorithm, first observe that we can ignore those choices of X for which the set Y_X has size $> k - |X|$. Therefore, the FPT algorithm we use to compute a minimum vertex cover of $\texttt{torso}(T, V \setminus X)$ runs in time $\mathcal{O}^*(1.2738^{k-|X|})$. Summing over

all choices of X, the time taken by our algorithm is upper bounded by

$$\begin{aligned}
&\sum_{x=0}^{k} \binom{n-k}{x} \mathcal{O}^*(1.2738^{k-x}) \\
&= \mathcal{O}^*\Big(1.2738^k \sum_{x=0}^{k} \binom{n-k}{x} \Big(\frac{1}{1.2738}\Big)^x\Big) \\
&= \mathcal{O}^*\Big(1.2738^k \Big(1+\frac{1}{1.2738}\Big)^{n-k}\Big) \\
&= \mathcal{O}^*\left(1.7851^n \Big(\frac{1.2738}{1.7851}\Big)^k\right)
\end{aligned}$$

This completes the proof of the lemma. □

Now we are ready to prove the main theorem of this section.

Theorem 3.5. *There is an algorithm that given an instance $(G = (V, E), T)$ of* Unrestricted Node Multiway Cut *returns an optimum solution in time $\mathcal{O}^*(1.4767^n)$, where $n = |V|$.*

Proof. Let (G, T) be a given instance of Unrestricted Node Multiway Cut and $|T| = k$. Recall that we have described two different algorithms for Unrestricted Node Multiway Cut. We now choose either of these algorithms based on the values of k and n. If $k \leq 0.5622n$, then we use the algorithm described in Lemma 3.2. In this case, the running time is upper bounded by $\mathcal{O}^*(2^k) \leq \mathcal{O}^*(2^{0.5622n}) \leq \mathcal{O}(1.4767^n)$. If $k > 0.5622n$, then we use the algorithm described in Lemma 3.4. This algorithm runs in time $\mathcal{O}^*\left(1.7850^n\big(\frac{1.2738}{1.7850}\big)^k\right)$ which is a decreasing function of k. Substituting $k = 0.5622n$, we get an upper bound on the running time as $\mathcal{O}^*(1.4767^n)$. This completes the proof of the theorem. □

3.2 Node Multiway Cut

In this subsection, we give an exact algorithm for the Node Multiway Cut problem. We start with the following observation which follows from the fact that any solution to an instance of Node Multiway Cut is disjoint from the set of terminals in the instance.

Observation 3.6. *Let (G, T) be an instance of* Node Multiway Cut. *If T is not an independent set in G, then there is no solution to the instance (G, T). Furthermore, if two terminals t_1 and t_2 have a common neighbor v, then v must be in every solution for the given instance.*

Due to Observation 3.6, we may assume that the terminal set is independent and the neighborhoods of the terminals in G are pairwise disjoint. This reduces the restricted Node Multiway Cut to the following generalization of the Unrestricted Node Multiway Cut, also known as the Group Multiway Cut problem.

Group Multiway Cut
Input: An undirected graph $G = (V, E)$ and pairwise disjoint sets of terminals $\{T_1, T_2, \ldots, T_\ell\}$.
Question: Find a set $S \subseteq V(G)$ of minimum size such that $G \setminus S$ has no $u - v$ path for any $u \in T_i, v \in T_j$ and $i \neq j$.

In the following, we describe an exact algorithm for Group Multiway Cut. The ideas and arguments are similar to those used in the proof of Theorem 3.5. First we give a structural lemma about the vertices in T_i.

Lemma 3.7. *Let $(G, T_1, \ldots, T_\ell)$ be an instance of* Group Multiway Cut. *Let G' be the graph obtained by removing from G, the edges in $G[T_i]$ for every $i \in \{1, \ldots, \ell\}$. Then, $S \subseteq V$ is a solution to the instance $(G, T_1, \ldots, T_\ell)$ if and only if it is also a solution to the instance $(G', T_1, \ldots, T_\ell)$.*

Proof. Since G' is a subgraph of G, any solution for $(G, T_1, \ldots, T_\ell)$ is also a solution for $(G', T_1, \ldots, T_\ell)$. We now consider the reverse direction. Let S be a solution for $(G', T_1, \ldots, T_\ell)$. We claim that S is also a solution for $(G, T_1, \ldots, T_\ell)$. Suppose that this is not the case, that is, there is a path P in G from $u_i \in T_i$ to $u_j \in T_j$ disjoint from S. Furthermore, we can assume without loss of generality that the internal vertices of P (if any) are disjoint from T_r for every $1 \leq r \leq \ell$. Observe that P is also present in the graph G', and therefore S intersects P by definition, a contradiction. □

Due to Lemma 3.7, henceforth we can assume that each T_i is an independent set in G. We have the following observation similar to Observation 3.1.

Observation 3.8. *Let $(G, T_1, \ldots, T_\ell)$ be an instance of* Group Multiway Cut. *Let S be an optimum solution to this instance and $\hat{T} = \cup_{i=1}^{\ell} T_i$. Then, $|S| \leq |\hat{T}|$.*

We have the following lemma which gives us an algorithm for the instances where $k = |\cup_{i=1}^{\ell} T_i|$ is "small". The proof of this lemma is very similar to the proof of Lemma 3.2.

Lemma 3.9. *Let $(G, T_1, \ldots, T_\ell)$ be an instance of* Group Multiway Cut, *let $\hat{T} = \cup_{i=1}^{\ell} T_i$ and let $|\hat{T}| = k$. Then we can find an optimum solution to the given instance of* Group Multiway Cut *in time $\mathcal{O}^*(2^k)$.*

Proof. We first reduce this instance to an instance of Node Multiway Cut, by creating a new terminal t_i, for each set T_i, which is then made adjacent to all vertices in T_i. We then apply the FPT algorithm of Cygan et al [10] for Node Multiway Cut to obtain the required solution. This takes time $\mathcal{O}^*(2^k)$. □

Next, we describe another algorithm for Group Multiway Cut. We will use this algorithm for those instances where $k = |\cup_{i=1}^{\ell} T_i|$ is "large". We have the following lemma, whose proof is very similar to the proof of Lemma 3.3.

Lemma 3.10. *Let $(G, T_1, \ldots, T_\ell)$ be an instance of* Group Multiway Cut *where $G = (V, E)$ and let S be an optimum solution to this instance. Let $X = S \setminus \hat{T}$ and $Y = S \cap \hat{T}$ where $\hat{T} = \cup_{i=1}^{\ell} T_i$. Then, Y is a vertex cover of ℓ-`torso`$(T_1, \ldots, T_\ell, V \setminus X)$ and conversely, for any vertex cover Y' for the graph ℓ-`torso`$(T_1, \ldots, T_\ell, V \setminus X)$, the set $X \cup Y'$ is a solution to this instance.*

Proof. We first show that Y is indeed a vertex cover of $G' = \ell$-`torso`$(T_1 \ldots, T_\ell, V \setminus X)$. Let E' be the edge set of G'. Suppose that Y is not a vertex cover of G' and there is an edge $(t_i, t_j) \in E'$ which is not covered by Y. Observe that if $t_i \in T_{i'}$ and $t_j \in T_{j'}$ for $i' \neq j'$ then $(t_i, t_j) \notin E$ since it contradicts the assumption of S being a solution. Furthermore, if $i' = j'$, then $(t_i, t_j) \notin E'$ by definition of `torso`. Therefore, it must be the case that there is a path P between t_i and t_j in $G[V \setminus X]$ whose internal vertices are disjoint from T. Since this path is disjoint from both X and Y, it is also present in the graph $G \setminus S$, a contradiction. Hence, we conclude that Y is indeed a vertex cover of G'.

We now show that for any vertex cover Y' of G', the set $X \cup Y'$ is a solution to the instance $(G, T_1, \ldots, T_\ell)$ of Group Multiway Cut. Suppose to the contrary that there is a vertex cover

Y' of G' such that the set $S' = X \cup Y'$ is not a solution to the instance $(G, T_1, \ldots, T_\ell)$. That is, there is a t_i-t_j path in $G \setminus S'$ for some $t_i \in T_{i'}$ and $t_j \in T_{j'}$ where $i' \neq j'$. Observe that this implies the existence of a t_p-t_q path in $G \setminus S'$ for some $t_p \in T_{p'}$ and $t_q \in T_{q'}$ where $p' \neq q'$ such that the internal vertices of P (if any) are disjoint from $\hat{T} \cup S'$. Therefore the edge (t_p, t_q) is not covered by Y' in G', a contradiction. This completes the proof of the lemma. □

Next, using Lemma 3.10 we can show the following lemma.

Lemma 3.11. *There is an algorithm that given an instance $(G = (V, E), T_1, \ldots, T_\ell)$ of* Group Multiway Cut *returns an optimum solution in time $\mathcal{O}^*\left(1.7851^n \left(\frac{1.2738}{1.7851}\right)^k\right)$, where $k = |\cup_{i=1}^{\ell} T_i|$ and $n = |V|$.*

Proof. Let $\hat{T} = \cup_{i=1}^{\ell} T_i$. The description of the algorithm is as follows. For every $X \subseteq (V \setminus \hat{T})$ such that $|X| \leq k$, we construct the graph $G_X = \ell\text{-}\texttt{torso}(T_1, T_2, \ldots, T_\ell, V \setminus X)$ and compute a minimum vertex cover Y_X for G_X. We compute the minimum vertex cover by using the FPT algorithm of [18], which runs in time $\mathcal{O}^*(1.2738^\ell)$ where ℓ is the size of an optimum vertex cover. Finally, we return the set $X \cup Y_X$ which is a smallest solution over all choices of X. The correctness of this algorithm follows from Lemma 3.10.

In order to bound the running time of this algorithm, first observe that we can ignore those choices of X for which the set Y_X has size $> k - |X|$. Therefore, the FPT algorithm we use to compute a minimum vertex cover of $\ell\text{-}\texttt{torso}(T_1, T_2, \ldots, T_\ell, V \setminus X)$ runs in time $\mathcal{O}^*(1.2738^{k-|X|})$.

Summing over all choices of X, the time taken by our algorithm is

$$\begin{aligned}
&\sum_{x=0}^{k} \binom{n-k}{x} \mathcal{O}^*(1.2738^{k-x}) \\
&= \mathcal{O}^*\left(1.2738^k \sum_{x=0}^{k} \binom{n-k}{x} \left(\frac{1}{1.2738}\right)^x\right) \\
&= \mathcal{O}^*\left(1.2738^k)\left(1 + \frac{1}{1.2738}\right)^{n-k}\right) \\
&= \mathcal{O}^*\left(1.7851^n \left(\frac{1.2738}{1.7851}\right)^k\right)
\end{aligned}$$

This completes the proof of the lemma. □

Combining the algorithms from Lemma 3.9 and Lemma 3.11, we can show the next theorem.

Theorem 3.12. *There is an algorithm that, given an instance $(G = (V, E), T_1, \ldots, T_\ell)$ of* Group Multiway Cut*, runs in time $\mathcal{O}^*(1.4767^n)$ and returns an optimum solution, where $n = |V|$.*

Proof. Let $(G, T_1, \ldots, T_\ell)$ be the given instance of Group Multiway Cut and $\hat{T} = \cup_{i=1}^{\ell} T_i$ and $|\hat{T}| = k$. Recall that we have described two different algorithms for Group Multiway Cut. We now choose either of these algorithms based on the values of k and n. If $k \leq 0.5622n$, then we use the algorithm described in Lemma 3.9. In this case, the running time is upper bounded by $\mathcal{O}^*(2^k) \leq \mathcal{O}^*(1.4767^n)$. If $k > 0.5622n$, then we use the algorithm described in Lemma 3.11. This algorithm runs in time $\mathcal{O}^*\left(1.7851^n \left(\frac{1.2738}{1.7851}\right)^k\right)$ which is a decreasing function of k. Substituting $k = 0.5622n$, we get an upper bound on the running time as $\mathcal{O}^*(1.4767^n)$. This completes the proof of the theorem. □

The following theorem follows from Theorem 3.12 and Observation 3.6.

Theorem 3.13. *There is an algorithm that given an instance $(G = (V, E), T)$ of* Node Multiway Cut *returns an optimum solution in time $\mathcal{O}^*(1.4767^n)$, where $n = |V|$.*

4 Directed Unrestricted Node Multiway Cut

In this section, we consider the Directed Unrestricted Node Multiway Cut problem.

> **Directed Unrestricted Node Multiway Cut**
> *Input*: A directed graph $D = (V, A)$ and a set of terminals $T = \{t_1, t_2, \ldots, t_k\}$.
> *Question*: Find a set $S \subseteq V$ of minimum size such that $G \setminus S$ has no $t_i \to t_j$ path for any $i \neq j$.

Since we consider the version where the terminals can be deleted, we have the following observation.

Observation 4.1. *Let (G, T) be an instance of* Directed Unrestricted Node Multiway Cut *and S be an optimum solution to this instance . Then, $|S| \leq |T|$.*

The proof of the next lemma is identical to the proof of Lemma 3.3 and therefore, we do not repeat it.

Lemma 4.2. *Let (D, T) be an instance of* Directed Unrestricted Node Multiway Cut *where $D = (V, A)$ and let S be an optimum solution to this instance. Let $X = S \setminus T$ and $Y = S \cap T$. Then Y is a vertex cover of the graph $\texttt{torso}(T, V \setminus X)$. Conversely if Y' is any vertex cover of the graph $\texttt{torso}(T, V \setminus X)$, then the set $X \cup Y'$ is a solution to this instance.*

Note that $\texttt{torso}(T, V \setminus X)$ is a directed graph. So by a vertex cover of $\texttt{torso}(T, V \setminus X)$ we mean a vertex cover of the underlying undirected graph of $\texttt{torso}(T, V \setminus X)$. Now we describe our algorithm for Directed Unrestricted Node Multiway Cut.

Theorem 4.3. Directed Unrestricted Node Multiway Cut *can be solved in $\mathcal{O}^*(1.6181^n)$ time.*

Proof. The description of the algorithm is as follows. For every $X \subseteq (V \setminus T)$ such that $|X| \leq k$, we construct the graph $D_X = \texttt{torso}(T, V \setminus X)$ and compute a minimum vertex cover Y_X for D_X. Note that we can ignore those choices of X for which the set Y_X has size $> k - |X|$. We compute the minimum vertex cover by using the FPT algorithm of [18], which runs in time $\mathcal{O}^*(1.2738^\ell)$ where ℓ is the size of an optimum vertex cover. Finally, we return the set $X \cup Y_X$ which is a smallest solution over all choices of X. The correctness of this algorithm follows from Lemma 4.2. We now bound the running time $\mathcal{T}$ of our algorithm. For every choice of X we run the FPT algorithm for

vertex cover, which takes time $\mathcal{O}^*(1.2738^{k-|X|})$. Therefore we have,

$$\begin{aligned}
\mathcal{T} &= \sum_{x=0}^{k} \binom{n-k}{x} \mathcal{O}^*(1.2738^{k-x}) \\
&\leq \sum_{x=0}^{k} \binom{n-k}{x} \mathcal{O}^*(1.6181^{k-x}) \\
&= \mathcal{O}^*(1.6181^k) \sum_{x=0}^{k} \binom{n-k}{x} \left(\frac{1}{1.6181}\right)^x \\
&\leq \mathcal{O}^*\Big(1.6181^k\Big)\Big(\frac{1}{1.6181}+1\Big)^{n-k}\Big) \\
&= \mathcal{O}^*\Big(1.6181^k \times (1.6181)^{n-k}\Big) \\
&= \mathcal{O}^*(1.6181^n)
\end{aligned}$$

This completes the proof of the theorem. □

5 Subset Feedback Vertex Set

In this section we design an exact algorithm for Subset Feedback Vertex Set. We actually design two different algorithms for the problem, and then use these two algorithms to construct our final exact algorithm.

Let (G, T) be the given instance of Subset Feedback Vertex Set. Recall that we are allowed to pick terminal vertices into a solution. The following observation follows from the fact that the set of terminals itself is a solution.

Observation 5.1. *Let (G, T) be an instance of* Subset Feedback Vertex Set*, and let S be an optimum solution to this instance. Then $|S| \leq |T|$.*

Lemma 5.2. *1. There is an algorithm that given an instance $(G = (V, E), T)$ of* Subset Feedback Vertex Set *returns an optimum solution in time $\mathcal{O}^*\left(1.2611^n\big(\frac{3.619}{1.2611}\big)^k\right)$, where $k = |T|$ and $n = |V|$.*

2. There is an algorithm that given an instance $(G = (V, E), T)$ of Subset Feedback Vertex Set *returns an optimum solution in time $\mathcal{O}^*\left(2^n\big(\frac{1.7347}{2}\big)^k\right)$, where $k = |T|$ and $n = |V|$.*

Proof. For every $X \subseteq (V \setminus T)$ such that $|X| \leq k$, let T_X be the set of terminals t such that $G \setminus X$ contains a cycle passing through t which contains no other terminal vertex. Let G_X be the graph obtained from $G \setminus (X \cup T_X)$ by contracting every connected component of $G \setminus (T \cup X)$. Given X, we can compute both T_X and G_X in polynomial time. Let Y_X be a minimum feedback vertex set for G_X containing only vertices of T.

For the first algorithm, we compute Y_X in the following manner. We assign a weight of $k+1$ to the vertices not in T and 1 to the vertices in T. We then use an FPT algorithm to compute a minimum feedback vertex set of G_X of weight at most k. The current fastest deterministic FPT algorithm for checking if there is a feedback vertex set of weight p in a graph with integral weights is due to [19], who extend the algorithm of Kociumaka and Pilipczuk [20] to weighted graphs, and

it runs in time $\mathcal{O}^*(3.619^p)$. The current fastest randomized algorithm weighted feedback vertex set with integral weights is due to Cygan et al. [10], and it runs in time $\mathcal{O}^*(3^p)$.

For the second algorithm we compute Y_X by computing a maximum induced forest of G_X which contains all the non-terminal vertices and taking its complement. Let F be the set of all non-terminal vertices in G_X and let $q = |V(G_X)| - |F|$ be the number of terminal vertices in G_X. We use the $\mathcal{O}^*(1.7347^q)$ algorithm of Fomin and Villanger [3] on (G_X, F) and compute a maximum induced forest of G_X containing F.

Let $S_X = X \cup T_X \cup Y_X$. We compute S_X for every X and output the one with the smallest number of vertices as our solution.

Correctness. The correctness of both the algorithms follows from the following claims.

Claim 5.3. *Let S be an optimum solution to the given instance of* Subset Feedback Vertex Set *and let $X = S \setminus T$. Let T_X be the set of terminals t such that $G \setminus X$ contains a cycle passing through t which contains no other terminal vertices. Then $T_X \subseteq S$.*

Proof. Let $t \in T_X$ and C_t is a cycle passing through T in $G \setminus X$ which doesn't contain any other terminal vertex. If $t \notin S$, then S doesn't intersect C_t. This is a contradiction. Thus $T_X \subseteq S$. This completes the proof of this claim. □

The above claim shows the correctness of adding T_X to the solution. Once T_X is added to the solution then there are no cycles containing exactly one vertex from T. In this case, the following lemma shows that it suffices to compute a minimum feedback vertex set for the graph G_X.

Claim 5.4. *Let S be an optimum solution to the given instance of* Subset Feedback Vertex Set *and let $X = S \setminus T$ and $Y = S \cap T$. Furthermore, suppose that there are no cycles in $G \setminus X$ containing a unique vertex of T, i.e., perform the update $T \leftarrow T \setminus T_X$. Let G_X be obtained from $G \setminus X$ by contracting every connected component of $G \setminus (T \cup X)$. Then Y is a minimum feedback vertex set of G_X. Conversely if Y' is any feedback vertex set of G_X, then the set $X \cup Y'$ is a solution for the given instance of* Subset Feedback Vertex Set.

Proof. We first show that for any $W \subseteq T$, there is a cycle in $G_X \setminus W$ if and only if there is a cycle in $G \setminus (X \cup W)$ which passes through $T \setminus W$. The claim then follows by setting $W = Y$ and $W = Y'$. First, observe that there is a unique connected component H_u in $G \setminus (T \cup X)$ corresponding to each non-terminal vertex $u \in G_X$.

Now consider a cycle C in $G_X \setminus W$ and observe that all the terminal vertices in C lie in $T \setminus W$. We can replace each non-terminal vertex u in C with a path P_u in H_u to obtain a closed walk in G. However this closed walk is actually a cycle, since the paths $\{P_u\}$ are pairwise vertex disjoint (they belong to different connected components of $G \setminus (T \cup X)$). Thus we obtain a cycle in $G \setminus (X \cup W)$.

Conversely let C be a cycle in $G \setminus (X \cup W)$ that passes through a terminal $t \in T$. We assume that C visits each connected component of $G \setminus (T \cup X)$ at most once: suppose the cycle C visits the component H_u (corresponding to the non-terminal $u \in G_X$) of $G \setminus (T \cup X)$ at least twice. Let y, y' be the first, last[2] vertices of $C \cap H_u$. Since H_u is connected, there is a $y - y'$ path P' contained completely within H_u. We replace the $y - y'$ path in C which contained at least one other vertex of H_u (since C visited H_u at least twice) by P'. Hence (the modified) C visits H_u exactly once now. If there are any more connected components of $G \setminus (T \cup X)$ which are visited by C more than once, then we repeat the above process for each such connected component. The final modified version

[2]That is, exactly one of the two $y - y'$ paths in C contains vertices from H_u

of the cycle satisfies the desired condition that it visits each connected component of $G \setminus (T \cup X)$ at most once.

By our assumption that there are no cycles in $G \setminus X$ containing exactly one terminal from T, we know that every terminal in $T \setminus W$ has at most one edge to a connected component in $G \setminus (T \cup X)$. Let P_u be a maximal subpath of C which is contained in the connected component H_u of $G \setminus (T \cup X)$. Consider the closed walk C' obtained from C by contracting the maximal subpath P_u to the vertex u, for every u. Observe that C' is actually a cycle since no vertex is repeated in C', and C' is present in $G_X \setminus W$. This completes the proof of this claim. □

Running Time. Let $\mathcal{T}_1$ be the running time of the first algorithm and, $\mathcal{T}_2$ be the running time of the second algorithms. The following two claims establish the running times of both the algorithms.

Claim 5.5. $\mathcal{T}_1 = \mathcal{O}^*\Big(1.2611^n \times \big(\frac{3.619}{1.2611}\big)^k\Big)$.

Proof. For a set X the potential solution we output has size $|Y_X| + |X| + |T_X|$. We know that the optimum is at most $|T| = k$ since deleting T itself is a feasible solution. Hence if $|Y_X| > k - |X|$ then we have $|Y_X| + |X| + |T_X| \geq |Y_X| + |X| > k$, and hence we can ignore this choice of X as it will never lead to an optimum solution. Hence we have that

$$\begin{aligned}
\mathcal{T}_1 &= \sum_{x=0}^{k} \binom{n-k}{x} \mathcal{O}^*(3.619^{k-x}) \\
&= \mathcal{O}^*(3.619^k) \sum_{x=0}^{k} \binom{n-k}{x} \Big(\frac{1}{3.619}\Big)^x \\
&\leq \mathcal{O}^*\Big(3.619^k \times \Big(1 + \frac{1}{3.619}\Big)^{n-k}\Big) \\
&= \mathcal{O}^*\Big(3.619^k \times (1.2611)^{n-k}\Big) \\
&= \mathcal{O}^*\Big(1.2611^n \times \Big(\frac{3.619}{1.2611}\Big)^k\Big)
\end{aligned}$$

□

Claim 5.6. $\mathcal{T}_2 = \mathcal{O}^*\Big(2^n \times \big(\frac{1.7347}{2}\big)^k\Big)$.

Proof. Observe the number of terminals is $q = |T| - |T_X| \leq |T| = k$. Therefore the exact algorithm runs in time $\mathcal{O}^*(1.7347^k)$. Therefore we have,

$$\begin{aligned}
\mathcal{T}_2 &= \sum_{x=0}^{k} \binom{n-k}{x} \mathcal{O}^*(1.7347^k) \\
&= \mathcal{O}^*(1.7347^k) \sum_{x=0}^{k} \binom{n-k}{x} \\
&= \mathcal{O}^*\Big(1.7347^k \times 2^{n-k}\Big) \\
&= \mathcal{O}^*\Big(2^n \times \Big(\frac{1.7347}{2}\Big)^k\Big)
\end{aligned}$$

□

This completes the proof of the lemma. □

Theorem 5.7. *There is an deterministic (resp. randomized) algorithm that given an instance* $(G = (V, E), T)$ *of* Subset Feedback Vertex Set *returns an optimum solution in time* $\mathcal{O}^*(1.8932^n)$ *(resp.* $\mathcal{O}^*(1.8826^n)$*), where* $n = |V|$.

Proof. Let (G, T) be the given instance of Subset Feedback Vertex Set, where G is a graph on n vertices and $|T| = k$. Based on the values of n and k we run one of the two algorithms described above.

Note that $\mathcal{T}_1, \mathcal{T}_2$ are increasing, decreasing functions when $k \in [0, n]$. Setting $\mathcal{T}_1 = \mathcal{T}_2$ gives $k = 0.3855n$. If $k \leq 0.3855n$, then we run the first algorithm described in Lemma 5.2. The running time is upper bounded by $\mathcal{O}^*\left(1.2611^n \times \left(\frac{3.619}{1.2611}\right)^{0.3855n}\right) = \mathcal{O}^*(1.8932^n)$. Otherwise if $k > 0.3855n$, then we run the second algorithm described in Lemma 5.2. This algorithm runs in time

$$\mathcal{O}^*\left(2^n \times \left(\frac{1.7347}{2}\right)^k\right)$$

which is a decreasing function of k. Substituting $k = 0.3855n$, we get an upper bound of $\mathcal{O}^*(1.8932^n)$ on the running time in this case as well.

We obtain the improved running time of $\mathcal{O}^*(1.8826^n)$ for a randomized algorithm by using the randomized $\mathcal{O}^*(3^k)$ algorithm of [21] in Claim 5.5 (instead of the deterministic $\mathcal{O}^*(3.619^k)$ algorithm of [19]). □

5.1 Subset Feedback Vertex Set on Chordal Graphs

In this section we give an algorithm for Subset Feedback Vertex Set on chordal graphs which improves upon the previous best algorithm of [17], and is much simpler. The main difference between this algorithm and the algorithm for Subset Feedback Vertex Set described earlier is that we use a polynomial time algorithm to solve weighted Feedback Vertex Set on chordal graphs ([22, 23]), instead of an FPT or an exact algorithm. Recall that a graph is chordal if it does not contain any C_t (for $t \geq 4$) as an induced subgraph. It is easy to see from this definition that chordal graphs are closed under vertex deletions and edge contractions: neither of these operations create larger induced cycles.

We are now ready to prove the main theorem of this section:

Theorem 5.8. *There is an algorithm that, given an instance* $(G = (V, E), T)$ *of* Subset Feedback Vertex Set *on chordal graphs, returns an optimum solution in* $\mathcal{O}^*(1.6181^n)$ *time, where* $n = |V|$.

Proof. The algorithm is the same as the two algorithms described in Lemma 5.2 except that we use the polynomial time algorithm for Feedback Vertex Set on chordal graphs instead of the FPT or the exact exponential algorithm. For every choice of X, we compute T_X and G_X in polynomial time. Observe that the graph G_X is obtained from G by vertex deletions and edge contractions, implying that G_X is also a chordal graph. We assign weight 1 to each terminal vertex and weight $k + 1$ to each non-terminal vertex, and compute in polynomial time a minimum weight feedback vertex set Y_X of G_X. This can be done by using a polynomial time algorithm of Yannakakis and Gavril [23] that computes a maximum weight induced forest in a chordal graph. Alternatively, we can use an algorithm of Corneil and Fonlupt [22]. We now analyze the running time of our algorithm.

Let S be any optimum solution and let $X = S \setminus T$. Observe that $|X| \leq |S| \leq |T|$. We now bound the number of choices for X

$$\begin{aligned}
\text{number of choices for } X &\leq \sum_{x=0}^{k} \binom{n-k}{x} && \text{(since } |T| = k\text{)} \\
&\leq \sum_{x=0}^{k} \binom{n-k}{x} \mathcal{O}^*(1.6181^{k-x}) && \text{(since } 1.6181 > 1\text{)} \\
&= \mathcal{O}^*(1.6181^k) \sum_{x=0}^{k} \binom{n-k}{x} \mathcal{O}^*(1.6181^{-x}) \\
&\leq \mathcal{O}^*(1.6181^k)\Big(1 + \frac{1}{1.6181}\Big)^{n-k} && \text{(by Binomial theorem)} \\
&= \mathcal{O}^*(1.6181^k)(1.6181)^{n-k} && \text{(since 1.6181 is a root of } x^2 - x - 1 = 0\text{)} \\
&= \mathcal{O}^*(1.6181^n)
\end{aligned}$$

Since after choosing X we do only a polynomial time computation, the running time of our algorithm is $\mathcal{O}^*(1.6181^n)$. □

We remark that we can use the above method to obtain faster exact algorithm for Subset Feedback Vertex Set on other graph classes, such as AT-free graphs [24], which are closed under vertex deletions and edge contractions, and Feedback Vertex Set is solvable in polynomial time on them.

6 Directed Subset Feedback Vertex Set

In this section we give an exact algorithm for the Directed Subset Feedback Vertex Set problem running in time $\mathcal{O}^*(1.9993^n)$. The problem is defined as follows.

> **Directed Subset Feedback Vertex Set**
> *Input*: A directed graph $D = (V, A)$ and a set of terminal vertices T of size k.
> *Question*: Find a minimum set of vertices in D which intersects every cycle in D which contains at least one vertex of T.

Next we observe the following property of directed graphs.

Observation 6.1. *Let $D = (V, A)$ be a directed graph. For any vertex $v \in V$, the following holds: v belongs to a closed walk in D if and only if v belongs to a cycle in D.*

Lemma 6.2. *Let $(D = (V, A), T)$ be an instance of* Directed Subset Feedback Vertex Set. *Let S be an optimum solution to this instance and $X = S \setminus T$, $Y = S \cap T$. Furthermore, suppose that every cycle in $D \setminus X$ that intersects T, contains at least two vertices of T. Then Y is a directed feedback vertex set in the graph $\texttt{torso}(T, V \setminus X)$ if and only if $X \cup Y$ is a directed subset feedback vertex set for the instance (D, T).*

Proof. Suppose $X \cup Y$ is a solution in D where $X \cap T = \emptyset$ and $Y \subseteq T$. If Y is not a directed feedback vertex set in $D_X = \texttt{torso}(T, V \setminus X)$, then there is a cycle C_X in $D_X \setminus Y$. From C_X in D_X we can obtain a closed walk C' in D in the following manner. We replace every edge (t_i, t_j) of C_X which is not present in A, with a path P_{ij} from t_i to t_j in $D \setminus X$ whose internal vertices lie

in $V \setminus (T \cup X)$. Therefore we get a closed walk C' in $D \setminus (X \cup Y)$ which contains a terminal. By Observation 6.1, there is a cycle in D which passes through a terminal in D, which is not covered by $X \cup Y$. This is a contradiction.

Conversely suppose that Y be a directed feedback vertex set in D_X, but $X \cup Y$ is not a solution for (D, T). Then there is a cycle C in $D \setminus (X \cup Y)$ that contains at least two vertices of T. Further assume that C is the shortest such cycle. Observe that every subpath P_{ij} of C from terminals t_i to t_j whose internal vertices lie in $V \setminus T$, implies an edge (t_i, t_j) in D_X. Therefore we can obtain a cycle C' in D_X from C by replacing the subpath P_{ij} with the edge (t_i, t_j), for every pair of terminals t_i, t_j in C. Observe that this cycle is not covered by Y. This is a contradiction.

This completes the proof of the lemma. □

The following observation is immediate since the set T forms a potential solution.

Observation 6.3. *Let (D, T) be an instance of* Directed Subset Feedback Vertex Set *and let S be an optimum solution for this instance. Then, $|S| \leq |T|$.*

We are now ready to prove the main theorem of this section.

Theorem 6.4. *There is an algorithm that given an instance $(D = (V, A), T)$ of* Directed Subset Feedback Vertex Set *returns an optimum solution in time $\mathcal{O}^*(1.9993^n)$, where $n = |V|$.*

Proof. The description of the algorithm is as follows. For every $X \subseteq (V \setminus T)$ such that $|X| \leq k$, we first compute (in polynomial time) the set T_X which is the set of terminals $t \in T$ such that there is a directed cycle in the graph $D[(V \setminus (T \cup X)) \cup \{t\}]$. Clearly T_X must be included in every optimum solution. In polynomial time, construct the graph $D_X = \texttt{torso}(T \setminus T_X, V \setminus X)$. Then we compute a minimum directed feedback vertex set Y_X for D_X by using the exact exponential algorithm by Razgon [16] for Directed Feedback Vertex Set, which runs in time $\mathcal{O}(1.9977^\ell)$ where $\ell = |V(D_X)| = |T \setminus T_X| \leq |T| = k$. Finally, we return the set $X \cup T_X \cup Y_X$ which is a smallest solution over all choices of X. The correctness of the algorithm follows from Lemma 6.2.

Running Time. The running time $\mathcal{T}$ of our algorithm is upper bounded by

$$\mathcal{T} \leq \sum_{x=0}^{k} \binom{n-k}{x} \mathcal{O}^*(1.9977^k).$$

To compute an upper bound on $\mathcal{T}$ only as a function of n, we need to examine the values of n and k. The following claim analyzes the running time of our algorithm, and completes the proof of the theorem.

Claim 6.5. $\mathcal{T} = \mathcal{O}(1.9993^n)$.

Proof. Let $\epsilon > 0$ be a constant which we fix later. Based on the values taken by n and k we consider the following two cases:

Case (1.) $k \leq \dfrac{n-k}{2+\epsilon}$. To address this case, we require the following well known result (see for example Lemma 3.13 in [1]).

Claim 6.6. *Let $0 < \alpha < 1$. Then*

$$\binom{n}{\alpha n} = \mathcal{O}^*\Big(\Big(\frac{1}{\alpha^\alpha (1-\alpha)^{1-\alpha}}\Big)^n\Big).$$

By Claim 6.6 for every $0 \leq x \leq k$ we have

$$\begin{aligned}
\binom{n-k}{x} &\leq \binom{n-k}{k} && (\text{since } k < \tfrac{n-k}{2}) \\
&\leq \binom{n-k}{\frac{n-k}{2+\epsilon}} && (\text{since } k < \tfrac{n-k}{2}) \\
&= \mathcal{O}^*\Big(\frac{2+\epsilon}{(1+\epsilon)^{\frac{1+\epsilon}{2+\epsilon}}}\Big)^{n-k} && (\text{by Claim 6.6})
\end{aligned}$$

The first restriction we impose on ϵ is that

$$\Big(\frac{2+\epsilon}{(1+\epsilon)^{\frac{1+\epsilon}{2+\epsilon}}}\Big) \geq 1.9977 \tag{1}$$

So, in this case we have

$$\mathcal{T} \leq \mathcal{O}^*\Big(1.9977^k \cdot \Big(\frac{2+\epsilon}{(1+\epsilon)^{\frac{1+\epsilon}{2+\epsilon}}}\Big)^{n-k}\Big) = \mathcal{O}^*\Big(\frac{2+\epsilon}{(1+\epsilon)^{\frac{1+\epsilon}{2+\epsilon}}}\Big)^{n} \tag{2}$$

Case (2.) $k > \dfrac{n-k}{2+\epsilon}$. In this case we have

$$\begin{aligned}
\mathcal{T} &= \sum_{x=0}^{k} \binom{n-k}{x} \mathcal{O}^*(1.9977^k) \\
&= \mathcal{O}^*(1.9977^k) \cdot \sum_{x=0}^{k} \binom{n-k}{x} \\
&\leq \mathcal{O}^*(1.9977^k \cdot 2^{n-k}) && (\text{by Binomial theorem}) \\
&= \mathcal{O}^*\Big(2^n \cdot \Big(\frac{1.9977}{2}\Big)^k\Big) \\
&\leq \mathcal{O}^*\Big(2^n \cdot \Big(\frac{1.9977}{2}\Big)^{\frac{n}{3+\epsilon}}\Big) && (\text{since } k > \tfrac{n-k}{2+\epsilon})
\end{aligned}$$

Equating the upper bound from Case (2.) with that from Equation 2 gives $\epsilon = 0.0565$. Note that this value of ϵ also satisfies Equation 1. Hence, we have that

$$\mathcal{T} = \mathcal{O}^*\Big(2^n \cdot \Big(\frac{1.9977}{2}\Big)^{\frac{n}{3.0565}}\Big) = \mathcal{O}^*(1.9993^n)$$

□

This completes the proof of the theorem. □

7 Conclusion

We introduced a methodology of obtaining non-trivial exact exponential algorithms for several *terminal set problems*. We conclude with open problems which seems to be evasive to our approach. Designing an algorithm faster than $\mathcal{O}^*(2^n)$ for Directed Node Multiway Cut remains an interesting question. Another interesting problem is Subset Odd Cycle Transversal, where

the task is to find a vertex subset of minimum size hitting all cycles of odd length containing at least one terminal. Again, the problem is trivially solvable in $\mathcal{O}^*(2^n)$ but no faster algorithm for this problem is known. We conclude by remarking that an approach based on our methodology might result in such an algorithm since Odd Cycle Transversal is solvable in time $\mathcal{O}^*(1.4661^n)$ [4, 5]. Finally designing an algorithm for Multicut on directed graphs, faster than the trivial $\mathcal{O}^*(2^n)$ algorithm, remains an interesting open problem. Only recently, it has been settled for undirected graphs. In particular, Lokshtanov et al. [25] obtained an algorithm for Multicut on undirected graphs running in time $\mathcal{O}^*(1.987^n)$.

References

[1] F. V. Fomin, D. Kratsch, Exact Exponential Algorithms, 1st Edition, Springer-Verlag New York, Inc., New York, NY, USA, 2010.

[2] M. Xiao, H. Nagamochi, Exact algorithms for maximum independent set, in: Algorithms and Computation - 24th International Symposium, ISAAC 2013, Hong Kong, China, December 16-18, 2013, Proceedings, 2013, pp. 328–338.

[3] F. V. Fomin, Y. Villanger, Finding induced subgraphs via minimal triangulations, in: STACS, Vol. 5, 2010, pp. 383–394.

[4] S. Mishra, V. Raman, S. Saurabh, S. Sikdar, König deletion sets and vertex covers above the matching size, in: ISAAC, Vol. 5369, 2008, pp. 836–847.

[5] J. M. Robson, Algorithms for maximum independent sets, J. Algorithms 7 (3) (1986) 425–440.

[6] S. Gupta, V. Raman, S. Saurabh, Maximum r-regular induced subgraph problem: Fast exponential algorithms and combinatorial bounds, SIAM J. Discrete Math. 26 (4) (2012) 1758–1780.

[7] M. Pilipczuk, M. Pilipczuk, Finding a maximum induced degenerate subgraph faster than 2^n, in: Parameterized and Exact Computation - 7th International Symposium, IPEC 2012, Ljubljana, Slovenia, September 12-14, 2012. Proceedings, 2012, pp. 3–12.

[8] W. Mader, Über die Maximalzahl kreuzungsfreier H-Wege, Arch. Math. (Basel) 31 (4) (1978/79) 387–402.

[9] J. Chen, Y. Liu, S. Lu, An Improved Parameterized Algorithm for the Minimum Node Multiway Cut Problem, Algorithmica 55 (1) (2009) 1–13.

[10] M. Cygan, M. Pilipczuk, M. Pilipczuk, J. O. Wojtaszczyk, On multiway cut parameterized above lower bounds, TOCT 5 (1) (2013) 3.

[11] N. Garg, V. V. Vazirani, M. Yannakakis, Multiway cuts in node weighted graphs, J. Algorithms 50 (1) (2004) 49–61.

[12] D. Marx, Parameterized Graph Separation Problems, Theor. Comput. Sci. 351 (3) (2006) 394–406.

[13] F. V. Fomin, P. Heggernes, D. Kratsch, C. Papadopoulos, Y. Villanger, Enumerating minimal subset feedback vertex sets, Algorithmica 69 (1) (2014) 216–231.

[14] I. Razgon, Exact computation of maximum induced forest, in: Algorithm Theory - SWAT 2006, 10th ScandinavianWorkshop on Algorithm Theory, Riga, Latvia, July 6-8, 2006, Proceedings, 2006, pp. 160–171.

[15] F. V. Fomin, S. Gaspers, A. V. Pyatkin, I. Razgon, On the minimum feedback vertex set problem: Exact and enumeration algorithms, Algorithmica 52 (2) (2008) 293–307.

[16] I. Razgon, Computing Minimum Directed Feedback Vertex Set in $O^*(1.9977^n)$, in: Theoretical Computer Science, 10th Italian Conference, ICTCS 2007, Rome, Italy, October 3-5, 2007, Proceedings, 2007, pp. 70–81.

[17] P. A. Golovach, P. Heggernes, D. Kratsch, R. Saei, Subset feedback vertex sets in chordal graphs, J. Discrete Algorithms 26 (2014) 7–15.

[18] J. Chen, I. A. Kanj, G. Xia, Improved upper bounds for vertex cover, Theor. Comput. Sci. 411 (40-42) (2010) 3736–3756.

[19] A. Agrawal, S. Kolay, D. Lokshtanov, S. Saurabh, A faster fpt algorithm and a smaller kernel for block graph vertex deletion, in: Latin American Symposium on Theoretical Informatics, Springer, 2016, pp. 1–13.

[20] T. Kociumaka, M. Pilipczuk, Faster deterministic feedback vertex set, Inf. Process. Lett. 114 (10) (2014) 556–560.

[21] M. Cygan, J. Nederlof, M. Pilipczuk, M. Pilipczuk, J. M. M. van Rooij, J. O. Wojtaszczyk, Solving connectivity problems parameterized by treewidth in single exponential time, in: IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011, 2011, pp. 150–159.

[22] D. Corneil, J. Fonlupt, The complexity of generalized clique covering, Discrete Applied Mathematics 22 (2) (1988–1989) 109 – 118.

[23] M. Yannakakis, F. Gavril, The maximum k-colorable subgraph problem for chordal graphs, Inf. Process. Lett. 24 (2) (1987) 133–137.

[24] D. Kratsch, H. Müller, I. Todinca, Feedback vertex set on AT-free graphs, Discrete Applied Mathematics 156 (10) (2008) 1936–1947.

[25] D. Lokshtanov, S. Saurabh, O. Suchý, Solving multicut faster than 2^n, in: Algorithms - ESA 2014 - 22th Annual European Symposium, Wroclaw, Poland, September 8-10, 2014. Proceedings, Vol. 8737 of Lecture Notes in Computer Science, Springer, 2014, pp. 666–676.