Journal of Computer and System Sciences 88 (2017) 195–207

Contents lists available at ScienceDirect

Journal of Computer and System Sciences

www.elsevier.com/locate/jcss

Faster exact algorithms for some terminal set problems [☆]

Rajesh Chitnis [a,1], Fedor V. Fomin [b], Daniel Lokshtanov [b], Pranabendu Misra [c,2], M.S. Ramanujan [b,*,3], Saket Saurabh [b,c]

[a] *Faculty of Mathematics and Computer Science, Weizmann Institute of Science, Israel*
[b] *Department of Informatics, University of Bergen, Norway*
[c] *The Institute of Mathematical Sciences, HBNI, Chennai, India*

A R T I C L E   I N F O

Article history:
Received 17 August 2015
Received in revised form 29 September 2016
Accepted 12 April 2017
Available online 4 May 2017

Keywords:
Exact exponential time algorithms
Terminal set problems
Multiway cut
Feedback vertex set

A B S T R A C T

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.[4] 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 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 $\mathcal{G}$?

☆ 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.

* Corresponding author.

E-mail addresses: rajesh.chitnis@weizmann.ac.il (R. Chitnis), fomin@ii.uib.no (F.V. Fomin), daniello@ii.uib.no (D. Lokshtanov), pranabendu@imsc.res.in (P. Misra), ramanujan@ac.tuwien.ac.at (M.S. Ramanujan), saket@imsc.res.in (S. Saurabh).

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

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

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

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

http://dx.doi.org/10.1016/j.jcss.2017.04.003
0022-0000/© 2017 Elsevier Inc. All rights reserved.

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 [9–12]. 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 $t_i \rightarrow t_j$ 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.