Edge Bipartization Faster Than 2^{k*}

Marcin Pilipczuk[1], Michał Pilipczuk[2], and Marcin Wrochna[3]

1 **Institute of Informatics, University of Warsaw, Poland**
malcin@mimuw.edu.pl
2 **Institute of Informatics, University of Warsaw, Poland**
michal.pilipczuk@mimuw.edu.pl
3 **Institute of Informatics, University of Warsaw, Poland**
m.wrochna@mimuw.edu.pl

Abstract

In the Edge Bipartization problem one is given an undirected graph G and an integer k, and the question is whether k edges can be deleted from G so that it becomes bipartite. In 2006, Guo et al. [6] proposed an algorithm solving this problem in time $\mathcal{O}(2^k \cdot m^2)$; today, this algorithm is a textbook example of an application of the iterative compression technique. Despite extensive progress in the understanding of the parameterized complexity of graph separation problems in the recent years, no significant improvement upon this result has been yet reported.

We present an algorithm for Edge Bipartization that works in time $\mathcal{O}(1.977^k \cdot nm)$, which is the first algorithm with the running time dependence on the parameter better than 2^k. To this end, we combine the general iterative compression strategy of Guo et al. [6], the technique proposed by Wahlström [18] of using a polynomial-time solvable relaxation in the form of a Valued Constraint Satisfaction Problem to guide a bounded-depth branching algorithm, together with an involved Measure&Conquer analysis of the recursion tree.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases edge bipartization, FPT algorithm

Digital Object Identifier 10.4230/LIPIcs.IPEC.2016.26

1 Introduction

The Edge Bipartization problem asks, for a given graph G and integer k, whether one can turn G into a bipartite graph using at most k edge deletions. Together with its close relative Odd Cycle Transversal (OCT), where one deletes vertices instead of edges, Edge Bipartization was one of the first problems shown to admit a fixed-parameter (FPT) algorithm using the technique of *iterative compression*. In a breakthrough paper [17] that introduces this methodology, Reed et al. showed how to solve OCT in time $\mathcal{O}(3^k \cdot kmn)$[1]. In fact, this was the first FPT algorithm for OCT. Following this, Guo et al. [6] applied iterative compression to show fixed-parameter tractability of several closely related problems,

* Mi. Pilipczuk and M. Wrochna have been supported by the Polish National Science Centre grant DEC-2013/11/D/ST6/03073. Mi. Pilipczuk has been supported by Foundation for Polish Science via the START stipend program. During the work on these results, Mi. Pilipczuk has been holding a post-doc position of Warsaw Centre of Mathematics and Computer Science. Ma. Pilipczuk has been supported by the Centre for Discrete Mathematics and its Applications (DIMAP) at the University of Warwick and by Warwick-QMUL Alliance in Advances in Discrete Mathematics and its Applications.

[1] Even though Reed et al. [17] state their running time as $\mathcal{O}(4^k \cdot kmn)$, it is not hard to adjust the analysis to show that the algorithm in fact works in time $\mathcal{O}(3^k \cdot kmn)$; see e.g. [7, 15].

including an algorithm for Edge Bipartization with running time $\mathcal{O}(2^k \cdot m^2)$. Today, both results are textbook examples of the iterative compression technique.

Iterative compression is in fact a simple idea that boils down to an algorithmic usage of induction. In case of Edge Bipartization, we introduce edges of G one by one, and during this process we would like to maintain a solution F to the problem, i.e., $F \subseteq E(G)$ is such that $|F| \leq k$ and $G - F$ is bipartite. When the next edge e is introduced to the graph, we observe that $F \cup \{e\}$ is a solution of size at most $k + 1$, that is, at most one too large. Then the task reduces to solving Edge Bipartization Compression: given a solution that exceeds the budget by at most one, we are asked to find a solution that fits into the budget.

Surprisingly, this simple idea leads to great algorithmic gains, as it reduces the matter to a cut problem. Guo et al. [6] showed that a simple manipulation of the instance reduces Edge Bipartization Compression to the following problem that we call Terminal Separation: We are given an undirected graph G with a set $\mathcal{T}$ of $k + 1$ disjoint pairs of terminals, where each terminal is of degree 1 in G. The question is whether one can color one terminal of every pair white and the second black in such a way that the minimum edge cut between white and black terminals is at most k. Thus, the algorithm of Guo et al. [6] boils down to trying all the 2^{k+1} colorings of terminals and solving a minimum edge cut problem. For OCT, we similarly have a too large solution $X \subseteq V(G)$ of size $k + 1$, and we are looking for a partition of X into (L, R, Z), where the size of the minimum vertex cut between L and R in $G - Z$ is at most $k - |Z|$. Thus it suffices to solve 3^{k+1} instances of a flow problem.

The search for FPT algorithms for cut problems has been one of the leading directions in parameterized complexity in the recent years. Among these, Odd Cycle Transversal and Edge Bipartization play a central role; see for instance [6, 12, 14, 17] and references therein. Of particular importance is the work of Kratsch and Wahlström [12], who gave the first (randomized) polynomial kernelization algorithms for both problems. The main idea is to encode the cut problems that arise when applying iterative compression into a matroid with a representation that takes small space. The result sparked a line of further work on applying matroids in parameterized complexity.

Another thriving area in parameterized complexity is the *optimality program*, probably best defined by Marx in [16]. The goal of it is to systematically investigate the optimum complexity of algorithms for parameterized problem by proving possibly tight lower and upper bounds. For the lower bounds methodology, the standard complexity assumptions used are the *Exponential Time Hypothesis* (*ETH*) and the *Strong Exponential Time Hypothesis* (*SETH*). In the recent years, the optimality program has achieved a number of successes. For instance, under the assumption of SETH, we now know the precise bases of exponents for many classical problems parameterized by treewidth [13]. To explain the complexity of fundamental parameterized problems for which natural algorithms are based on dynamic programming on subsets, Cygan et al. [1] introduced a new hypothesis resembling SETH, called the *Set Cover Conjecture* (*SeCoCo*). See [13, 16] for more examples.

For our techniques, the most important is the line of work of Guillemot [5], Cygan et al. [3], Lokshtanov et al. [14], and Wahlström [18] that developed a technique for designing parameterized algorithm for cut problems called *LP-guided branching*. The idea is to use the optimum solution to the linear programming (LP) relaxation of the considered problem in order to measure progress. Namely, during the construction of a candidate solution by means of a backtracking process, the algorithm achieves progress not only when the budget for the size of the solution decreases (as is usual in branching algorithms), but also when the LP lower bound on the optimum solution increases. Using this concept, Cygan et al. [3] showed a $2^k n^{\mathcal{O}(1)}$-time algorithm for Node Multiway Cut. Lokshtanov et al. [14] further

refined this technique and applied it to improve the running times of algorithms for several important cut problems. In particular, they obtained a $2.315^k n^{\mathcal{O}(1)}$-time algorithm for Odd Cycle Transversal, which was the first improvement upon the classic $\mathcal{O}(3^k \cdot kmn)$-time algorithm of Reed et al. [17]. From the point of view of the optimality program, this showed that the base 3 of the exponent was not the final answer for Odd Cycle Transversal.

In [3, 14] it was essential that the considered LP relaxation is half-integral, which restricts the applicability of the technique. Recently, Wahlström [18] proposed to use stronger relaxations in the form of certain polynomial-time solvable Valued Constraint Satisfaction Problems (VCSPs). Using this idea, he showed efficient FPT algorithms for node and edge deletion variants of Unique Label Cover, for which natural LP relaxations are not half-integral.

Despite substantial progress on the node deletion variant, for Edge Bipartization there has been no improvement since the classic algorithm of Guo et al. [6] that runs in time $\mathcal{O}(2^k \cdot m^2)$. The main technical contribution of Lokshtanov et al. [14] is a $2.315^k n^{\mathcal{O}(1)}$-time algorithm for Vertex Cover parameterized by the excess above the value of the LP relaxation (VC-above-LP); the algorithm for OCT then follows from folklore reductions from OCT to VC-above-LP via the Almost 2-SAT problem. Thus the algorithm for OCT in fact relies on the LP relaxation for Vertex Cover, which has very strong combinatorial properties; in particular, it is half-integral. No such strong and simple relaxation is available for Edge Bipartization. The natural question stemming from the optimality program, whether the 2^k term for Edge Bipartization can be improved, was asked repeatedly in the parameterized complexity community, e.g. by Daniel Lokshtanov at WorKer'13 [2].

Our results and techniques. In this paper we answer this question in affirmative:

▶ **Theorem 1.1.** Edge Bipartization *can be solved in time* $\mathcal{O}(1.977^k \cdot nm)$.

To prove this, we begin with the approach of Guo et al. [6], using iterative compression to reduce solving Edge Bipartization to solving Terminal Separation (see Section 2 for a formal definition of the latter). This problem has two natural parameters: $|\mathcal{T}|$, the number of terminal pairs, and p, the bound on the size of the cut between white and black terminals. The approach of Guo et al. is to use a simple $\mathcal{O}(2^{|\mathcal{T}|} \cdot pm)$ algorithm that tries all colorings of terminal pairs and computes the size of a minimum cut between the colors.

The observation that is crucial to our approach is that one can express Terminal Separation as a very restricted instance of the Edge Unique Label Cover problem. More precisely, in this setting the task is to assign each vertex of G a label from $\{\mathbf{A}, \mathbf{B}\}$. Pairs of $\mathcal{T}$ present hard (of infinite cost) inequality constraints between the labels of terminals involved, while edges of G present soft (of unit cost) equality constraints between the endpoints. The goal is to minimize the cost of the labeling, i.e., the number of soft constraints broken. An application of the results of Wahlström [18] (with further improvements of Iwata, Wahlström, and Yoshida [8] regarding linear dependency on the input size) immediately gives an $\mathcal{O}(4^p \cdot m)$ algorithm for Terminal Separation.

Thus, we have in hand two substantially different algorithms for Terminal Separation. If we plug in $|\mathcal{T}| = k + 1$ and $p = k$, as is the case in the instance that we obtain from Edge Bipartization Compression, then we obtain running times $\mathcal{O}(2^k \cdot km)$ and $\mathcal{O}(4^k \cdot m)$, respectively. The idea now is that these two algorithms present two complementary approaches to the problem, and we would like to combine them to solve the problem more efficiently. To this end, we need to explain more about the approach of Wahlström [18].

The algorithm of Wahlström [18] is based on measuring the progress by means of the optimum solution to the relaxation of the problem (in the form of a Valued CSP instance).

In our case, this relaxation of Terminal Separation has the following form: We assign each vertex a label from $\{\bot, \mathbf{A}, \mathbf{B}\}$, where $\bot$ is an additional marker that should be thought of as *not yet decided*. The hard constraints have zero cost only for labelings $(\mathbf{A}, \mathbf{B})$, $(\mathbf{B}, \mathbf{A})$ and $(\bot, \bot)$, and infinite cost otherwise. The soft constraints have cost 0 for equal labels on the endpoints, 1 for unequal from $\{\mathbf{A}, \mathbf{B}\}$, and $\frac{1}{2}$ when exactly one endpoint is assigned $\bot$. Based on previous results of Kolmogorov, Thapper, and Živný [11], Wahlström observed that this relaxation is polynomial-time solvable, and moreover it is *persistent*: whenever the relaxation assigns $\mathbf{A}$ or $\mathbf{B}$ to some vertex, then it is safe to perform the same assignment in the integral problem (i.e., not relaxed, only with the "integral" labels $\mathbf{A}, \mathbf{B}$). The algorithm constructs an integral labeling with a backtracking process that fixes labels of consecutive vertices of the graph. During this process, it maintains an optimum solution to the relaxation that is moreover *maximal*, in the sense that one cannot extend the current labeling by fixing integral labels on some undecided vertices without increasing the cost. This can be done by dint of persistence and polynomial-time solvability: we can check in polynomial time whether a non-trivial extension exists, and then it is safe to fix the labels of vertices that get decided. Thus, when the algorithm considers the next vertex u and branches into two cases, fixing label $\mathbf{A}$ or $\mathbf{B}$ on it, the optimum cost of the relaxation increases by at least $\frac{1}{2}$ in each branch. Hence the recursion tree can be pruned at depth $2p$, and we obtain a $4^p n^{\mathcal{O}(1)}$-time algorithm.

Our algorithm for Terminal Separation applies a similar branching strategy, where at each point we maintain some labeling of the vertices with $\mathbf{A}$, $\mathbf{B}$, and $\bot$ (undecided). Every terminal pair is either already *resolved* (assigned $(\mathbf{A}, \mathbf{B})$ or $(\mathbf{B}, \mathbf{A})$), or *unresolved* (assigned $(\bot, \bot)$). Using the insight of Wahlström we can assume that this labeling is maximal. Intuitively, we look at unresolved pairs from $\mathcal{T}$ and try to identify a pair (s, t) for which branching into labelings $(\mathbf{A}, \mathbf{B})$ and $(\mathbf{B}, \mathbf{A})$ leads to substantial progress. Here, we measure progress in terms of a potential μ that is a linear combination of three components:

- t, the number of unresolved terminal pairs;
- k, the current budget for the cost of the sought integral solution;
- ν, the difference between k and the cost of the current solution to the relaxation.

These ingredients are taken with weights $\alpha_t = 0.59950$, $\alpha_\nu = 0.29774$, and $\alpha_k = 1 - \alpha_t - \alpha_\nu = 0.10276$. Thus, the largest weight is put on the progress measured in terms of the number of resolved terminal pairs: We want to argue that if we can identify a possibility of recursing into two instances, where in each of them at least one new terminal pair gets resolved, but in one of them we resolve two terminal pairs, then we can pursue this branching step.

Therefore, we are left with the following situation: when branching on any terminal pair, only this terminal pair gets resolved in both branches. Then the idea is to find a branching step where the decrease of the auxiliary components of the potential, namely ν and k, is significant enough to ensure the promised running time of the algorithm. Here we apply an extensive combinatorial analysis of the instance to show that finding such a branching step is always possible. In particular, our analysis can end up with a branching not on a terminal pair, but on the label of some other vertex; however, we make sure that in both branches some terminal pair gets eventually resolved. Also, in some cases we localize a part of the input that can be simplified (a *reduction step*), and then the analysis is restarted.

To sum up, we would like to highlight two aspects of our contribution. First, we answer a natural question stemming from the optimality program, showing that 2^k is not the final dependency on the parameter for Edge Bipartization. Second, our algorithm can be seen as a "proof of concept" that the LP-guided branching technique, even in the more abstract variant of Wahlström [18], can be combined with involved Measure&Conquer analysis of the branching tree. Note that in the past Measure&Conquer and related techniques led to rapid progress in the area of moderately-exponential algorithms [4].

We remark that the goal of the current paper is clearly improving the 2^k term, and not optimizing the dependence of the running time on the input size. However, we do estimate it. Using the tools prepared by Iwata, Wahlström, and Yoishida [8], we are able to implement the algorithm so that it runs in time $\mathcal{O}(1.977^k \cdot nm)$. Naively, this seems like an improvement over the algorithm of Guo et al. [6] that had quadratic dependence on m, however this is not the case. We namely use the recent approximation algorithm for Edge Bipartization of Kolay et al. [9] that in time $\mathcal{O}(k^{\mathcal{O}(1)} \cdot m)$ either returns a solution F^{apx} of size at most $\mathcal{O}(k^2)$, or correctly concludes that there is no solution of size k. Then we start iterative compression from $G - F^{\text{apx}}$ and introduce edges of F^{apx} one by one, so we need to solve the Terminal Separation problem only $\mathcal{O}(k^2)$ times. In our case each iteration takes time $\mathcal{O}(1.977^k \cdot nm)$, but for the approach of Guo et al. it would take time $\mathcal{O}(2^k \cdot km)$. Thus, by using the same idea based on [9], the algorithm of Guo et al. can be adjusted to run in time $\mathcal{O}(2^k \cdot k^3 m)$.

2 Overview of the algorithm

As announced in the introduction, the application of iterative compression and the reduction to a Terminal Separation instance closely follows the approach of [6]; in this extended abstract, we give only an overview of the branching algorithm for Terminal Separation.

Let us start with some notation. Consider a graph G with a family $\mathcal{T}$ of disjoint pairs of vertices in G; we call those vertices *terminals*. A *terminal separation* is a pair (A, B) with $A, B \subseteq V(G)$ such that $A \cap B = \emptyset$ and, for every terminal pair P, either one of the terminals in P belongs to A and the second to B, or $P \subseteq V(G) \setminus (A \cup B)$. A terminal separation (A, B) is *integral* if $A \cup B = V(G)$.[2] A terminal separation (A', B') *extends* (A, B) if $A \subseteq A'$ and $B \subseteq B'$. The *cost* of a terminal separation (A, B) is defined as $c(A, B) = (d(A) + d(B))/2$, where $d(X)$ is the number of edges between X and $V(G) \setminus X$, for $X \subseteq V(G)$. Note that if (A, B) is integral, then we have $c(A, B) = d(A) = d(B)$. We say that a terminal separation (A, B) is *maximal* if every other separation extending it has strictly larger cost.

Terminal Separation
Input: A graph G with a set of disjoint terminal pairs $\mathcal{T}$ such that every terminal is of degree at most one in G; a terminal separation (A°, B°); and an integer k.
Goal: Find an integral terminal separation (A, B) extending (A°, B°) of cost at most k, or report that no such separation exists.

We borrow the basic toolbox from [18, 8], in the form of the following two statements.

▶ **Theorem 2.1** (persistence [18]). *Let $(G, \mathcal{T}, (A^\circ, B^\circ), k)$ be a* Terminal Separation *instance, and let (A, B) be a terminal separation in G of minimum cost among separations that extend (A°, B°). Then there exists an integral separation (A^*, B^*) that has minimum cost among all integral separations extending (A°, B°), with the additional property that (A^*, B^*) extends (A, B).*

▶ **Theorem 2.2** (polynomial-time solvability, [18, 8]). *Given a* Terminal Separation *instance $(G, \mathcal{T}, (A^\circ, B^\circ), k)$ with $c(A^\circ, B^\circ) \leq k$, one can in $\mathcal{O}(k^{\mathcal{O}(1)}m)$ time find a maximal terminal separation (A, B) in G that has minimum cost among all separations extending (A°, B°).*

[2] The word *integral* stems from the fact that an integral separation corresponds to a solution to the relaxed Terminal Separation problem that actually does not use the relaxed value $\perp$. In fact, it also corresponds to an integral solution of an LP formulation underlying the algorithmic results of [11].

From Theorems 2.1 and 2.2 it follows that, while working on a Terminal Separation instance $(G, \mathcal{T}, (A^\circ, B^\circ), k)$, we can always assume that (A°, B°) is a maximal separation: If that is not the case, we can obtain an extending separation (A, B) via Theorem 2.2, and set $(A^\circ, B^\circ) := (A, B)$; the safeness of the last step is guaranteed by Theorem 2.1.

2.1 The potential to measure progress of the algorithm

Let $\mathcal{I} = (G, \mathcal{T}, (A^\circ, B^\circ), k)$ be a Terminal Separation instance, where (A_0, B_0) is a maximal terminal separation; we henceforth call such an instance *maximal*. We are interested in keeping track of the following partial measures:

- $t_\mathcal{I}$ is the number of unresolved terminal pairs;
- $\nu_\mathcal{I} = k - c(A^\circ, B^\circ)$;
- $k_\mathcal{I} = k$.

The $\mathcal{O}(2^k km)$-time algorithm used in [6] can be interpreted in our framework as an $\mathcal{O}(2^{t_\mathcal{I}} k_\mathcal{I} m)$-time algorithm for Terminal Separation, while the generic LP-branching algorithm for Edge Unique Label Cover of [18, 8] can be interpreted as an $\mathcal{O}(4^{\nu_\mathcal{I}} m)$-time algorithm. Our main goal is to blend the two, by analyzing the cases where both perform badly.

An important insight is that all these inefficient cases happen when A° and B° increase their common boundary. If this is the case, a simple reduction rule is applicable that also reduces the allowed budget k.

▶ **Reduction 2.3** (Boundary Reduction). *If there exists an edge ab with $a \in A^\circ$, $b \in B^\circ$, then delete the edge ab and decrease k by one. If there exist two edges va, vb with $a \in A^\circ$, $b \in B^\circ$, and $v \notin A^\circ \cup B^\circ$, then delete both edges va and vb, and decrease k by one.*

In some sense, with this reduction rule the budget k represents the yet undetermined part of the boundary between A^* and B^* in the final integral solution (A^*, B^*). For this reason, we also include the budget k in the potential.

Formally, we fix three constants $\alpha_t = 0.59950$, $\alpha_\nu = 0.29774$, and $\alpha_k = 1 - \alpha_t - \alpha_\nu = 0.10276$ and define a potential of an instance $\mathcal{I}$ as

$$\mu_\mathcal{I} = \alpha_t \cdot t_\mathcal{I} + \alpha_\nu \cdot \nu_\mathcal{I} + \alpha_k \cdot k_\mathcal{I}.$$

Our main technical result is the following.

▶ **Theorem 2.4.** *A Terminal Separation instance $\mathcal{I}$ can be solved in time $\mathcal{O}(c^{\mu_\mathcal{I}} nm)$ for some $c < 1.977$.*

We remark that instances of Terminal Separation we encounter while solving an Edge Bipartization instance (G, k) satisfy $t_\mathcal{I} = k + 1$, $\nu_\mathcal{I} = k$, and $\alpha_k = k$, hence $\mu_\mathcal{I} < k + 1$. Consequently, Theorem 1.1 follows from Theorem 2.4 by using it in the general iterative compression approach proposed by Guo et al. [6].

The algorithm of Theorem 2.4 follows a typical outline of a recursive branching algorithm. At every step, the current instance is analyzed, and either it is reduced, or some two-way branching step is performed. The potential $\mu_\mathcal{I}$ is used to measure the progress of the algorithm and to limit the size of the branching tree.

Observe that the Boundary Reduction reduces already determined parts of the boundary between A^* and B^* for the minimum-cost solution (A^*, B^*), and hence the integer $k_\mathcal{I}$, present in the potential $\mu_\mathcal{I}$, represents the yet unknown part of this boundary. It is easy to see that every application of the Boundary Reduction decreases the potential by exactly α_k; in multiple branches we show that a sufficient number of Boundary Reductions follow the branching step to ensure the promised running time bound.

2.2 Structure of a branching step

In every branching step, we identify two terminal separations (A_1, B_1) and (A_2, B_2) extending (A°, B°), and branch into two subcases; in subcase i we replace (A°, B°) with (A_i, B_i). We always argue the *correctness* of a branch by showing that there exists an integral solution (A^*, B^*) extending (A°, B°) of minimum cost, with the additional property that (A^*, B^*) extends (A_i, B_i) for some $i = 1, 2$. In subcase i, we apply the algorithm of Theorem 2.2 to $(G, \mathcal{T}, (A_i, B_i), k)$ to obtain a maximal separation (A_i°, B_i°), and pass the instance $\mathcal{I}_i = (G, \mathcal{T}, (A_i^\circ, B_i^\circ), k)$ to a recursive call.

To show the *running time bound* for a branching step, we analyze how the measure $\mu_\mathcal{I}$ decreases in the subcases, taking into account the reductions performed in the subsequent recursive calls. More formally, we say that a branching case *fulfills a branching vector* $[t_1, \nu_1, k_1;\ t_2, \nu_2, k_2]$ if, in subcase $i = 1, 2$, at least t_i terminal pairs become resolved or reduced with one of the reductions, the cost of the separation (A_i°, B_i°) grows by at least $\nu_i/2$, and the Boundary Reduction gets applied at least k_i times in the instance $(G, \mathcal{T}, (A_i^\circ, B_i^\circ), k)$.

A branching vector $[t_1, \nu_1, k_1; t_2, \nu_2, k_2]$ is *good* if

$$1.977^{-\alpha_t t_1 - \alpha_\nu \nu_1/2 - \alpha_k k_1} + 1.977^{-\alpha_t t_2 - \alpha_\nu \nu_2/2 - \alpha_k k_2} < 1.$$

Standard arguments for branching algorithms show that, if in every case we perform a branching step that fulfills some good branching vector, the branching tree originated from an instance $\mathcal{I}$ has $\mathcal{O}(c^{\mu_\mathcal{I}})$ leaves for some $c < 1.977$. To simplify further exposition, we gather in the next lemma good branching vectors used in the analysis; the fact that they are good can be checked by direct calculations.

▶ **Lemma 2.5.** *The following branching vectors are good:*

$[1, 1, 0; 2, 1, 0]$	$[1, 1, 1; 1, 2, 3]$	$[1, 2, 0; 1, 3, 1]$	$[1, 1, 0; 1, 4, 3]$	$[1, 1, 2; 1, 2, 2]$
$[1, 1, 1; 1, 3, 2]$	$[1, 3, 0; 1, 3, 0]$	$[1, 1, 0; 1, 5, 2]$	$[1, 2, 1; 1, 2, 2]$	$[1, 1, 1; 1, 4, 1]$

Let us stop here to comment that the vectors in Lemma 2.5 explain our choice of constants α_t, α_ν, α_k. The constant α_t is sufficiently large to make the vector $[1, 1, 0; 2, 1, 0]$ good; intuitively speaking, we are always done when in one branch we manage to resolve or reduce at least two terminal pairs. The choice of α_ν and α_k represents a very delicate tradeoff that makes both $[1, 1, 1; 1, 2, 3]$ and $[1, 2, 0; 1, 3, 1]$ good; note that setting $\alpha_\nu = 1 - \alpha_t$ and $\alpha_k = 0$ makes the first vector not good, while setting $\alpha_\nu = 0$ and $\alpha_k = 1 - \alpha_t$ makes the second vector not good. Arguably, the possibility of a tradeoff that makes both the second and the third vector of Lemma 2.5 good at the same time is one of the critical insights in our work.

2.3 Low-excess sets

A set $A \subseteq V(G)$ is an A°-*extension* if $A^\circ \subseteq A \subseteq V(G) \setminus B^\circ$. It is *terminal-free* if $A \setminus A^\circ$ does not contain any terminal. We denote by $\Delta(A) := d(A) - d(A^\circ)$ the *excess* of an A°-extension A. An A°-extension A is *compact* if $A \setminus A^\circ$ is connected and $E(A \setminus A^\circ, A^\circ) \neq \emptyset$.

One of the main technical tools for analysis is the study of extensions of small excess. We show that their structure can be reduced to have a relatively simple picture. While in this section we focus on supersets of the set A°, by symmetry the same conclusion holds if we swap the roles of A° and B°.

First, since (A°, B°) is maximal, we have that A° is the only terminal-free A°-extension of nonpositive excess. As for excess 1, one can show the following.

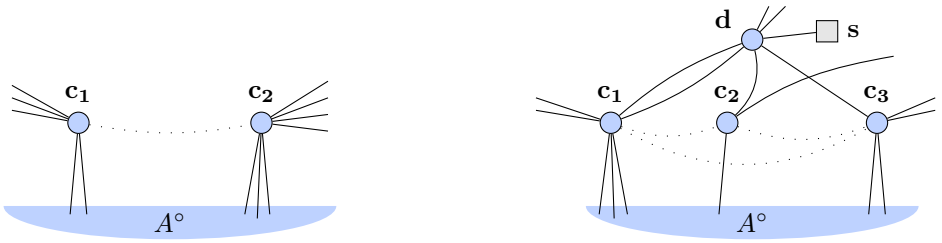

Figure 1 Examples of sets of excess 2 after reductions (dotted lines are non-edges). On the right a strict (non-null) extension A_s of $A^\circ \cup \{s\}$ with excess 1 is shown. For any such extension, $A_s \setminus \{s\}$ is a set of excess 2 in which the vertex d, obtained from contracting the set D of the decomposition, is the only neighbor of the terminal s.

▶ **Lemma 2.6.** *If A is a terminal-free A°-extension of excess 1, then there exists a minimum cost integral terminal separation (A^*, B^*) extending (A°, B°), such that $(A \setminus A^\circ)$ is either completely contained in A^* or completely contained in B^*.*

Hence, one can collapse into a single vertex the set $A \setminus A^\circ$ for every terminal-free A°-extension A of excess 1. For extensions of excess 2, one can describe them similarly, and collapse into a single vertex any set D as in the following lemma.

▶ **Lemma 2.7.** *Assume that every terminal-free A°-extension of excess 1 has been collapsed to a single vertex, and that G contains no nonterminal degree-1 vertices. If A is a terminal-free A°-extension of excess 2, then there exists a partition $A \setminus A^\circ = D \uplus C_1 \uplus C_2 \uplus \ldots \uplus C_r$ for some $r \geq 0$ ($\uplus$ meaning union of disjoint sets), such that:*

1. *there exists a minimum cost integral terminal separation (A^*, B^*) extending (A°, B°), such that one of the following holds:*
   - *$(A \setminus A^\circ) \cap A^* = \emptyset$;*
   - *$(A \setminus A^\circ) \cap A^* = C_i$ for some $1 \leq i \leq r$; or*
   - *$A \subseteq A^*$.*
2. *for every $1 \leq i \leq r$, the sets C_i and $E(C_i, A^\circ)$ are nonempty, and $A^\circ \cup C_i$ is a terminal-free A°-extension of excess 1;*
3. *if $D \neq \emptyset$, then for every $1 \leq i \leq r$ the set $E(C_i, D)$ is nonempty and $A \setminus A^\circ$ is connected;*
4. *if $D = \emptyset$, then $r = 2$;*
5. *for every $1 \leq i < j \leq r$, there are no edges between C_i and C_j.*

2.4 Basic branching step

Let $\mathcal{T}' := \mathcal{T} \setminus (A^\circ \cup B^\circ)$ be the set of unresolved terminal pairs. In the basic branching step of our algorithm we take a pair $\{s, t\} \in \mathcal{T}'$ and try to assign s and t to the different sides of the separation. That is, we apply the algorithm of Theorem 2.2 twice: once for terminal separation $(A^\circ \cup \{s\}, B^\circ \cup \{t\})$, and the second time for terminal separation $(A^\circ \cup \{t\}, B^\circ \cup \{s\})$. In this manner we obtain two maximal terminal separations (A_s, B_t) and (A_t, B_s) that extend $(A^\circ \cup \{s\}, B^\circ \cup \{t\})$ and $(A^\circ \cup \{t\}, B^\circ \cup \{s\})$ respectively. Of course, the number of unresolved pairs decreases by at least one in both (A_s, B_t) and (A_t, B_s), due to resolving $\{s, t\}$.

If the number of unresolved pairs either in (A_s, B_t) or in (A_t, B_s) decreases by more than one, then performing a branching step $(A_1, B_1) = (A_s, B_t)$ and $(A_2, B_2) = (A_t, B_s)$ leads to the branching vector $[1, 1, 0; 2, 1, 0]$ or a better one, which is good; the corresponding decrease in the measure $\nu_{\mathcal{I}}$ follows from the assumption that (A°, B°) is maximal. We can test in

$\mathcal{O}(k^{\mathcal{O}(1)}m)$ time whether this holds for any pair $\{s,t\} \in \mathcal{T}'$, and if so then we pursue the branching step.

If this is not the case, we are left with the extensive analysis of the sets A_s, B_s, A_t, and B_t. As we could always pick $A_s = A^\circ \cup \{s\}$, and similarly for the other sets, we have that the excess of any of these four sets is at most one. Furthermore, the maximality of (A°, B°) implies that also neither of these excesses is negative: if, say, $\Delta(A_s) < 0$, then since $\Delta(B_t) \leq 1$, we have $c(A_s, B_t) \leq c(A^\circ, B^\circ)$, contradicting the maximality of (A°, B°). Thus, we are left with excesses 0 and 1, giving different cases for analysis.

Let us first consider the case when $A_s = A^\circ \cup \{s\}$, $A_t = A^\circ \cup \{t\}$, $B_s = B^\circ \cup \{s\}$, and $A_t = B^\circ \cup \{t\}$, that is, the situation when both branching steps colored only the terminals. If s or t is an isolated vertex in G, it is easy to reduce the pair $\{s,t\}$ without branching. Otherwise, let s' be the unique neighbor of s and t' be the unique neighbor of t. Since both s and t are of degree one in G, it is easy to argue that there exists a minimum-cost solution (A^*, B^*) that does not cut the edge ss'. Consequently, we can strengthen the basic branch by forcing s' to be in the same side of the separation as s. More precisely, we consider branches $(A_{ss'\to A}, B_{ss'\to A})$ and $(A_{ss'\to B}, B_{ss'\to B})$ that are minimum-cost terminal separations extending $(A^\circ \cup \{s,s'\}, B^\circ \cup \{t\})$ and $(A^\circ \cup \{t\}, B^\circ \cup \{s,s'\})$, computed using Theorem 2.2. It is easy to see that, unless some simple reduction is applicable or another terminal pair gets resolved, we have $\Delta(A_{ss'\to A}) \geq 2$ and $\Delta(B_{ss'\to B}) \geq 2$ while still $\Delta(B_{ss'\to A}), \Delta(A_{ss'\to B}) \geq 1$. This gives a good branching vector $[1,3,0;1,3,0]$.

In the analysis of remaining cases we rely on our understanding of low-excess extensions in the following way. Assume that, say, A_s is a strict superset of $A^\circ \cup \{s\}$. Then A_s contains s' and $A := A_s \setminus \{s\}$ is a terminal free A°-extension of excess $\Delta(A_s) + 1 \in \{1,2\}$. Thus, Lemma 2.6 or 2.7 applies, giving us a good insight into the set $A \setminus A^\circ$, capturing the neighborhood of s. Observe that the structure of an excess-1 or excess-2 extension in particular guarantees that A lies "closely" to the set A°, giving grounds for possibly multiple Boundary Reductions in a subcase in a branching step when some vertices of A are assigned to the B-side of the separation.

An extensive case analysis, provided in the full version of the paper, shows that in all cases, if the basic branching resolves only one terminal pair, then one can gather a sufficient number of Boundary Reductions stemming from the understanding of low-excess extensions and sufficient increase in the cost of the separation (A°, B°) to obtain a good branching vector. This proves Theorem 2.4. In the remainder of this section, we illustrate how the low-excess extensions work by sketching one particular subcase of the case $\Delta(A_s) = 1$ and $\Delta(B_s) = 0$. This illustration is quite representative for the kind of reasoning we need to perform in other cases as well.

2.5 Example subcase of the case $\Delta(A_s) = 1$, $\Delta(B_s) = 0$

We define $R = V(G) \setminus (A_s \cup B_s)$, $\tilde{A} = A_s \setminus B_s$ and $\tilde{B} = B_s \setminus A_s$; note that $\tilde{A}$ and $\tilde{B}$ are terminal-free extensions of A° and B°, respectively. By posimodularity of the cuts we infer:

$$d(A_s) + d(B_s) = d(\tilde{A}) + d(\tilde{B}) + 2|E(A_s \cap B_s, R)| \geq d(A^\circ) + d(B^\circ) + 2|E(A_s \cap B_s, R)|.$$

We infer that in our case $|E(A_s \cap B_s), R)| = 0$ and $\Delta(\tilde{A}) + \Delta(\tilde{B}) = 1$. In this overview we consider the subcase $\Delta(\tilde{A}) = 1$ and $\Delta(\tilde{B}) = 0$.

By maximality of (A°, B°) we have $\tilde{B} = B^\circ$; by Lemma 2.6 we can assume $\tilde{A} = A^\circ \cup \{a\}$ for some vertex a; in particular $A_s \supsetneq A^\circ \cup \{s\}$. Let s' be the unique neighbor of s; we have $s' \in A_s$, as otherwise $A_s \setminus \{s\}$ is a nontrivial A°-extension of excess 0, a contradiction. As $\Delta(A_s) = 1$, the set $A_s \setminus \{s\}$ is a terminal-free A°-extension of excess 2: we can apply Lemma 2.7

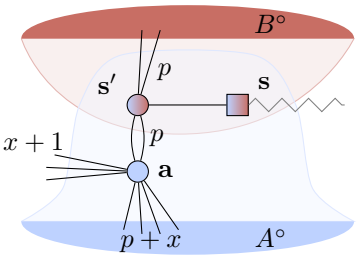

Figure 2 Subcase $(\Delta(A_s), \Delta(B_s)) = (\Delta(\tilde{A}), \Delta(\tilde{B})) = (1, 0)$. Extensions A_s, B_s are highlighted.

to obtain a decomposition $A_s \setminus \{s\} = A^\circ \uplus \{d, c_1, c_2, \ldots, c_r\}$ or $A_s \setminus \{s\} = A^\circ \uplus \{c_1, c_2\}$ (vertices d and c_i are sets D and C_i from Lemma 2.7 collapsed into single vertices due to reduction rules). From the fact that $s' \in A_s$ and (A_s, B_t) is a separation extending $(A^\circ \cup \{s\}, B^\circ \cup \{t\})$ of minimum-cost, we infer that s' is actually the vertex d: if $s' = c_i$ for some i, then $(A^\circ \cup \{s, c_i\}, B_t)$ would have strictly smaller cost. In particular, we are dealing with the decomposition of the form $A_s \setminus \{s\} = A^\circ \uplus \{d, c_1, c_2, \ldots, c_r\}$.

Since $\Delta(B_s) = 0$ but $\Delta(B^\circ \cup \{s\}) = 1$, we infer that $B_s \supsetneq B^\circ \cup \{s\}$, which implies that $s' \in B_s$. Furthermore, we can assume that $A_s \cap B_s = \{s, s'\}$: if there were more vertices in $A_s \cap B_s$, it is easy to see that we can safely reduce the graph by collapsing $(A_s \cap B_s) \setminus \{s\}$ into a single vertex. Consequently, as $\tilde{A} = A^\circ \cup \{a\}$ and $\tilde{B} = B^\circ$, we have $B_s = B^\circ \cup \{s, s'\}$ and $A_s = A^\circ \cup \{a, s', s\}$ (i.e., $r = 1$ and $c_1 = a$).

From Lemma 2.7 we have $p := |E(a, s')| \geq 1$ and $|E(a, A^\circ)| \geq 1$, thus $E(a, B^\circ) = \emptyset$ since Boundary Reduction does not apply to a. By using the assumptions on excesses of sets, we have that a is incident on: p edges to s', $x+1$ edges to $V(G) \setminus (A_s \cup B^\circ)$, $p+x$ edges to A° and no other edges, for some $x \geq 0$. Since $B_s = B^\circ \cup \{s', s\}$ is an excess-0 set and $E(s', R) = \emptyset$, we have that $|E(s', B^\circ)| = p + |E(s', A^\circ)|$. In particular $|E(s', B^\circ)| > 0$, so since Boundary Reductions do not apply to s', we have $E(s', A^\circ) = \emptyset$ and hence $|E(s', B^\circ)| = p$. See Fig. 2.

Consider first case $x = 0$. Then a has a unique edge aa' with $a' \in R$. If a' is a terminal, it is easy to either reduce the case without branching (if $a' = t$) or provide a branching step that resolves two terminal pairs (if a' belongs to other terminal pair), so assume otherwise. We claim that it is a safe reduction to contract the edge aa'; to prove this claim, it suffices to show that there exists an optimum integral terminal separation extending (A°, B°) where a and a' belong to the same side. Take any such integral terminal separation (A^*, B^*), and assume that a and a' are on opposite sides. Clearly it cannot happen that $a \in B^*$ and $a' \in A^*$, because then moving a from B^* to A^* would decrease the cost of the separation. Hence $a \in A^*$ and $a' \in B^*$. If $s' \in B^*$, then moving a from A^* to B^* would decrease the cost of the separation, so also $s' \in A^*$. Construct a new integral separation (A^*_m, B^*_m) from (A^*, B^*) by moving $\{a, s'\}$ from A^* to B^*. Then the cost of (A^*_m, B^*_m) is not larger than that of (A^*, B^*) (we could have broken the edge $s's$ instead of aa'), while both endpoints of aa' belong to A^*_m. This resolves the case $x = 0$.

In the case $x > 0$, we claim that branching on the membership of a leads to a good branch. That is, we recurse into two branches $(A_{a\to A}, B_{a\to A})$ and $(A_{a\to B}, B_{a\to B})$ that are minimum-cost maximal terminal separations extending $(A^\circ \cup \{a\}, B^\circ)$ and $(A^\circ, B^\circ \cup \{a\})$, respectively. For $X \in \{A, B\}$, let $t_{a\to X}, \nu_{a\to X}, k_{a\to X}$ be the changes of the components of the potential in respective branches, as we denote them in branching vectors.

Consider first the branch $(A_{a\to A}, B_{a\to A})$. Then p Boundary Reductions are triggered on vertex s' (regardless of whether it is added or not to one of the sets $A_{a\to A}, B_{a\to A}$). Hence $k_{a\to A} \geq p$. Moreover, the terminal pair $\{s, t\}$ either is already resolved by $(A_{a\to A}, B_{a\to A})$

or is easily reducible after applying the Boundary Reductions. Hence $t_{a\to A} \geq 1$. Finally, since (A°, B°) was maximal, we have that $\nu_{a\to A} \geq 1$. So the part of the branching vector corresponding to the branch $(A_{a\to A}, B_{a\to A})$ is $[1, 1, p]$, or better.

Consider now the second branch $(A_{a\to B}, B_{a\to B})$. Then at least $|E(a, A^\circ)| = p + x$ Boundary Reductions are triggered, hence $k_{a\to B} \geq p + x$. Since $p \geq 1$ and t is of degree 1, $s' \in B_{a\to B}$ and w.l.o.g. we can assume $s \in B_{a\to B}$ and $t \in A_{a\to B}$. Hence $t_{a\to B} \geq 1$. If actually $t_{a\to A} \geq 2$ or $t_{a\to B} \geq 2$, then we arrive at a good branching vector $[1, 1, p; 2, 1, p]$ or better, so assume that $t_{a\to A} = t_{a\to B} = 1$, that is, only the pair $\{s, t\}$ gets resolved.

We now claim that $\Delta(A_{a\to B}) \geq 1$ and $\Delta(B_{a\to B}) \geq 1$. For the latter claim, note that if $\Delta(B_{a\to B}) \leq 0$, then $B_{a\to B} \setminus \{s\}$ is a terminal-free B°-extension of excess at most one, while $a, s' \in B_{a\to B}$; a contradiction to the assumption that every terminal-free extension of excess one has been collapsed into a single vertex. For the former claim, suppose for the sake of contradiction that $d(A_{a\to B}) = d(A^\circ)$ (case $d(A_{a\to B}) < d(A^\circ)$ can easily be excluded by the maximality of (A°, B°)). Recall that also $d(B_s) = d(B^\circ)$, which means that $d(A_{a\to B}) + d(B_s) = c(A^\circ, B^\circ)$. From the posimodularity of cuts it now follows that one of the terminal separations $(A_{a\to B} \setminus B_s, B_s)$ and $(A_{a\to B}, B_s \setminus A_{a\to B})$ has cost not larger than (A°, B°), while both of them resolve the terminal pair $\{s, t\}$. This is a contradiction with the maximality of (A°, B°). Hence $\Delta(A_{a\to B}) \geq 1$ and $\Delta(B_{a\to B}) \geq 1$, and so $\nu_{a\to B} \geq 2$.

Thus, branching into separations $(A_{a\to A}, B_{a\to A})$ and $(A_{a\to B}, B_{a\to B})$ leads to a branching vector $[1, 1, p; 1, 2, p + x]$ or better. Recalling that $p, x > 0$, observe that this branching vector can be not good only if $p = x = 1$ and $\Delta(B_{a\to B}) = 1$. Let us now analyze this case.

Since $\Delta(B_{a\to B}) = 1$, we have that $B_{a\to B} \setminus \{s\}$ is a terminal-free set of excess 2, and hence we can apply Lemma 2.7 to it: assuming excess-2 sets have been reduced, we have that $B_{a\to B} \setminus \{s\}$ has a decomposition of the form $B^\circ \uplus \{c_1, c_2\}$ or $B^\circ \uplus \{d, c_1, \ldots, c_r\}$. Note that $B^\circ \cup \{s'\}$ is an excess-1 set, so it is not hard to argue that $s' = c_i$ for some i. As $a \in B_{a\to B}$, a is adjacent to s', and c_i-s are pairwise non-adjacent, we must have that $a = d$ and we are dealing with a decomposition of the form $B^\circ \uplus \{d, c_1, \ldots, c_r\}$. Observe that $B^\circ \cup \{a, s'\}$ is a B°-extension of excess at least $1 + (x + 1) = 3$ (counting edge ss' and edges between a and A°); hence $B_{a\to B} \supsetneq B^\circ \cup \{a, s', s\}$, and in particular $r > 1$. Hence there exists some vertex $c_j \neq c_i = s'$. By Lemma 2.7 we have that c_j is adjacent both to B° and to a. Hence, in the branch $(A_{a\to A}, B_{a\to A})$ at least one Boundary Reduction is applied to c_j, regardless whether c_j is assigned to $A_{a\to A}$, or $B_{a\to A}$, or neither of these sets. We did not include this Boundary Reduction in the previous calculations; this shows that we in fact pursue a branch with a branching vector $[1, 1, 2; 1, 2, 2]$ or better, which is a good branching vector.

3 Conclusions

In this work we have developed an algorithm for Edge Bipartization with running time $\mathcal{O}(1.977^k \cdot nm)$, which is the first one to achieve the dependence on the parameter better than 2^k. Thus, in the case of Edge Bipartization the constant 2 in the base of the exponent is not the ultimate answer, as is conjectured for CNF-SAT. Also, it improves some recent works where the FPT algorithm for Edge Bipartization is used as a black-box [10]. However, our work leaves some open questions that we would like to highlight.

- Reducing the dependence on the parameter from 2^k to 1.977^k can be only considered a "proof of concept" that such an improvement is possible. We put forward the question of designing a reasonably simple algorithm with significant improvement in the base of the exponent, hopefully decreasing the polynomial dependence from $\mathcal{O}(nm)$ to (near-)linear.
- Our approach can be summarized as follows: having observed that Terminal Separation admits a simple $\mathcal{O}^\star(2^{|\mathcal{T}|})$-time algorithm and an $\mathcal{O}^\star(4^k)$-time algorithm using the

CSP-guided technique of Wahlström [18], we develop an algorithm for a joint parameterization $(|\mathcal{T}|, k)$ that for $|\mathcal{T}| = k + 1$ achieves running time $\mathcal{O}^\star(1.977^k)$. Can Terminal Separation be solved in time $\mathcal{O}^\star(c^{|\mathcal{T}|})$ for some $c < 2$?

References

1 Marek Cygan, Holger Dell, Daniel Lokshtanov, Dániel Marx, Jesper Nederlof, Yoshio Okamoto, Ramamohan Paturi, Saket Saurabh, and Magnus Wahlström. On problems as hard as CNF-SAT. *ACM Trans. Algorithms*, 12(3):41, 2016. doi:10.1145/2925416.

2 Marek Cygan, Łukasz Kowalik, and Marcin Pilipczuk. Open problems from the update meeting on graph separation problems, Workshop on Kernels, Warsaw, 2013. http://worker2013.mimuw.edu.pl/slides/update-opl.pdf.

3 Marek Cygan, Marcin Pilipczuk, Michał Pilipczuk, and Jakub Onufry Wojtaszczyk. On Multiway Cut parameterized above lower bounds. *TOCT*, 5(1):3, 2013. doi:10.1145/2462896.2462899.

4 Fedor V. Fomin and Dieter Kratsch. *Exact Exponential Algorithms*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2010. doi:10.1007/978-3-642-16533-7.

5 Sylvain Guillemot. FPT algorithms for path-transversal and cycle-transversal problems. *Discrete Optimization*, 8(1):61–71, 2011. doi:10.1016/j.disopt.2010.05.003.

6 Jiong Guo, Jens Gramm, Falk Hüffner, Rolf Niedermeier, and Sebastian Wernicke. Compression-based fixed-parameter algorithms for feedback vertex set and edge bipartization. *J. Comput. Syst. Sci.*, 72(8):1386–1396, 2006.

7 Falk Hüffner. Algorithm engineering for optimal Graph Bipartization. *J. Graph Algorithms Appl.*, 13(2):77–98, 2009.

8 Yoichi Iwata, Magnus Wahlström, and Yuichi Yoshida. Half-integrality, lp-branching, and FPT algorithms. *SIAM J. Comput.*, 45(4):1377–1411, 2016. doi:10.1137/140962838.

9 Sudeshna Kolay, Pranabendu Misra, M. S. Ramanujan, and Saket Saurabh. Parameterized approximations via d-Skew-Symmetric Multicut. In *Proc. MFCS'14*, volume 8635 of *Lecture Notes in Computer Science*, pages 457–468. Springer, 2014. doi:10.1007/978-3-662-44465-8_39.

10 Sudeshna Kolay, Fahad Panolan, Venkatesh Raman, and Saket Saurabh. Parameterized algorithms on perfect graphs for deletion to (r, l)-graphs. In Piotr Faliszewski, Anca Muscholl, and Rolf Niedermeier, editors, *Proc. MFCS'14*, volume 58 of *LIPIcs*, pages 75:1–75:13. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2016. doi:10.4230/LIPIcs.MFCS.2016.75.

11 Vladimir Kolmogorov, Johan Thapper, and Stanislav Živný. The power of linear programming for general-valued CSPs. *SIAM J. Comput.*, 44(1):1–36, 2015.

12 Stefan Kratsch and Magnus Wahlström. Compression via matroids: A randomized polynomial kernel for Odd Cycle Transversal. *ACM Trans. Algorithms*, 10(4):20:1–20:15, 2014.

13 Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Lower bounds based on the Exponential Time Hypothesis. *Bulletin of the EATCS*, 105:41–72, 2011.

14 Daniel Lokshtanov, N. S. Narayanaswamy, Venkatesh Raman, M. S. Ramanujan, and Saket Saurabh. Faster parameterized algorithms using linear programming. *ACM Trans. Algorithms*, 11(2):15:1–15:31, 2014. doi:10.1145/2566616.

15 Daniel Lokshtanov, Saket Saurabh, and Somnath Sikdar. Simpler parameterized algorithm for OCT. In Jirí Fiala, Jan Kratochvíl, and Mirka Miller, editors, *Proc. IWOCA 2009, Revised Selected Papers*, volume 5874 of *Lecture Notes in Computer Science*, pages 380–384. Springer, 2009. doi:10.1007/978-3-642-10217-2_37.

16 Dániel Marx. What's next? Future directions in Parameterized Complexity. In *The Multivariate Algorithmic Revolution and Beyond*, volume 7370 of *Lecture Notes in Computer Science*, pages 469–496. Springer, 2012.

17 Bruce A. Reed, Kaleigh Smith, and Adrian Vetta. Finding odd cycle transversals. *Oper. Res. Lett.*, 32(4):299–301, 2004.

18 Magnus Wahlström. Half-integrality, LP-branching and FPT algorithms. In *Proc. SODA'14*, pages 1762–1781. SIAM, 2014.