Directed Flow-Augmentation

Eun Jung Kim*
eun-jung.kim@dauphine.fr
Université Paris-Dauphine, PSL Research University,
CNRS, UMR 7243, LAMSADE, 75016, Paris
Paris, France

Stefan Kratsch
kratsch@informatik.hu-berlin.de
Humboldt-Universität zu Berlin
Berlin, Germany

Marcin Pilipczuk†
m.pilipczuk@mimuw.edu.pl
University of Warsaw
Warsaw, Poland

Magnus Wahlström
Magnus.Wahlstrom@rhul.ac.uk
Royal Holloway, University of London, TW20 0EX
London, UK

ABSTRACT

We show a flow-augmentation algorithm in directed graphs: There exists a randomized polynomial-time algorithm that, given a directed graph G, two integers $s, t \in V(G)$, and an integer k, adds (randomly) to G a number of arcs such that for every minimal st-cut Z in G of size at most k, with probability $2^{-\mathrm{poly}(k)}$ the set Z becomes a *minimum* st-cut in the resulting graph.

The directed flow-augmentation tool allows us to prove (randomized) fixed-parameter tractability of a number of problems parameterized by the cardinality of the deletion set, whose parameterized complexity status was repeatedly posed as open problems:

(1) Chain SAT, defined by Chitnis, Egri, and Marx [ESA'13, Algorithmica'17],
(2) a number of weighted variants of classic directed cut problems, such as Weighted st-Cut, Weighted Directed Feedback Vertex Set, or Weighted Almost 2-SAT.

By proving that Chain SAT is (randomized) FPT, we confirm a conjecture of Chitnis, Egri, and Marx that, for any graph H, if the List H-Coloring problem is polynomial-time solvable, then the corresponding vertex-deletion problem is fixed-parameter tractable (with the remark that our algorithms are randomized).

CCS CONCEPTS

• **Theory of computation → Fixed parameter tractability**.

KEYWORDS

directed graphs, fixed-parameter tractability, flow-augmentation, Chain SAT

*Supported by the grant from French National Research Agency under JCJC program (ASSK: ANR-18-CE40-0025-01).

†This research is a part of a project that have received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme Grant Agreement 714704.

STOC '22, June 20–24, 2022, Rome, Italy

ACM ISBN 978-1-4503-9264-8/22/06...$15.00
https://doi.org/10.1145/3519935.3520018

ACM Reference Format:
Eun Jung Kim, Stefan Kratsch, Marcin Pilipczuk, and Magnus Wahlström. 2022. Directed Flow-Augmentation. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing (STOC '22), June 20–24, 2022, Rome, Italy.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3519935.3520018

1 INTRODUCTION

The study of graph separation problems has been one of the more vivid areas of parameterized complexity in the recent 10–15 years. The term "graph separation problems" captures a number of classic graph problems where, given a (undirected or directed) graph G with a cut budget k (and possibly some annotations, such as terminal vertices), one aims at obtaining some separation via at most k edge or vertex deletions. For example, the classic st-Cut problem asks to delete at most k edges so that there is no s-to-t path in the resulting graph and the Feedback Vertex Set asks to remove at most k vertices so that the resulting graph does not contain any cycles (i.e., is a forest in the undirected setting or a DAG in the directed setting). In all these problems, the cardinality of the deletion set is a natural parameter to study.

In 2004, Marx introduced the notion of *important separators* [15, 16] that turned out to be the key to fixed-parameter tractability of Multiway Cut and Directed Feedback Vertex Set [1], among many other examples. In subsequent years, the study of graph separation problems resulted in a rich toolbox of algorithmic techniques, such as *shadow removal* [19], treewidth reduction [17], or randomized contractions [2, 7, 8]. At SODA'21, the current authors added one more item to this list: a flow-augmentation technique in undirected graphs [10]. As a result of these developments, most open problems regarding parameterized complexity of graph separation problems in undirected graphs has been resolved. Our work [10] shows that this statement can be formalized in some sense: every problem that can be formalized as a Min UNSAT problem for some Boolean CSP language Γ that cannot express implication, is either $W[1]$-hard or FPT by one of the aforementioned techniques.

For directed graphs, the chartered landscape is much less complete. The notion of important separators and related technique of shadow removal generalizes to directed graphs, leading to fixed-parameter tractability of Directed Feedback Vertex Set [1], Directed Multiway Cut [6], and Directed Subset Feedback Vertex Set [4]. A number of problems whose undirected counterparts are FPT turned out to be $W[1]$-hard in the directed setting, including Directed Multicut [6, 20] or Directed Odd Cycle Transversal [14].

However, these results are far from satisfactorily charting the parameterized complexity of directed graph separation problems. Arguably, we seem to lack algorithmic techniques. Most notably, the powerful treewidth reduction theorem [17], stating that (in undirected graphs) all separations of size at most k between two fixed terminals live in a part of the graph with treewidth bounded by $2^{O(k)}$, seems not to have any meaningful counterpart in directed graphs. As a result, essentially all known FPT algorithms for graph separation problems in directed graphs rely in some part on important separators, which is a greedy argument bounding the number of cuts between two terminals of bounded size that have inclusion-wise maximal set of vertices reachable from one of the terminal. This "greedy" part of this argument breaks down whenever a problem at hand has some weights or annotations.

On the other hand, despite efforts in the last years, we were not able to prove lower bounds for FPT algorithms for many directed graph separation problems. This suggests that maybe there are still more algorithmic techniques to be explored, leading to more positive tractability results.

In this work, we provide such a technique: we show that flow-augmentation, introduced by current authors in [10] for undirected graphs, generalizes to directed graphs.

Theorem 1.1. *There exists a randomized polynomial-time algorithm that, given a directed graph G, two vertices $s, t \in V(G)$, and an integer k, outputs a set $A \subseteq V(G) \times V(G)$ such that the following holds: for every minimal st-cut $Z \subseteq E(G)$ of size at most k, with probability $2^{-O(k^4 \log k)}$ Z remains an st-cut in $G + A$ and, furthermore, Z is a* minimum *st-cut in G.*

Here, a set $Z \subseteq E(G)$ is an *st-cut* if there is no path from s to t in $G - Z$; it is *minimal* if no proper subset of Z is an st-cut and *minimum* if it is of minimum possible cardinality. By $G + A$ we mean the graph G with all elements of A added as infinity-capacity arcs.

The proof of Theorem 1.1 is sketched in Section 2. Full proof can be found in the arXiv version [9]. While the proof borrows a general outline and course of action from its undirected counterpart [10], at lower level most details are different, as directed graphs pose different challenges. The main differences are discussed in Section 2.

Applications. To illustrate the applicability of the directed flow-augmentation, let us first consider the Weighted st-Cut problem. Here, we are given a directed graph G with two terminals $s, t \in V(G)$, a weight function $\omega : E(G) \to \mathbb{Z}_+$, and two integers $k, W \in \mathbb{Z}_+$, and we ask for an st-cut Z of cardinality at most k and total weight at most W. This problem is known to be NP-hard and FPT when parameterized by $k + W$ [11].

By using directed flow-augmentation, we can ensure that the sought solution Z is actually a *minimum* st-cut (i.e., of minimum cardinality). Then, a solution can easily be found in polynomial-time: take $M := 1+\sum_{e\in E(G)} \omega(e)$, set the capacity of every edge e as $\omega(e) + M$, and find an st-cut of minimum capacity. By repeating the algorithm $2^{O(k^4 \log k)}$ times to ensure constant success probability, we obtain the following.

Theorem 1.2. *Weighted st-Cut can be solved in randomized time $2^{O(k^4 \log k)} n^{O(1)}$.*

The above approach turns out to be more general. An instance of Bundled Cut consists of a directed graph G, vertices $s, t \in V(G)$, a nonnegative integer k, and a family $\mathcal{B}$ of pairwise disjoint subsets of $E(G)$. The elements of $\mathcal{B}$ are henceforth called *bundles*. A *cut* in a Bundled Cut instance $\mathcal{I} = (G, s, t, k, \mathcal{B})$ is an st-cut Z with $Z \subseteq \bigcup \mathcal{B}$. The Bundled Cut problem asks for an st-cut that intersects at most k bundles, that is, $|\{B \in \mathcal{B} \mid Z \cap B \neq \emptyset\}| \leq k$. One can define a weighted variant of the Bundled Cut problem where every bundle $B \in \mathcal{B}$ is equipped with a weight $\omega(B) \in \mathbb{Z}_+$, we are given also a weight bound W, and we ask for an st-cut Z that intersects at most k bundles whose total weight is at most W.

If all bundles are singletons, then Bundled Cut just asks for an st-cut of size at most k (with the edges of $E(G) \setminus \bigcup \mathcal{B}$ being undeletable), so it is polynomial-time solvable. It is known that, when restricted to bundles of size 2, Bundled Cut is $W[1]$-hard when parameterized by k [18]. To get tractability, we define the following restriction called (Weighted) Bundled Cut with Order where we require the following: for every $B \in \mathcal{B}$ one can order the arcs of B as $e_1^B, \ldots, e_{|B|}^B$ such that for every $1 \leq i < j \leq |B|$ there exists a path $P_{i,j}^B$ in G that goes from one endpoint of e_i^B to one endpoint of e_j^B and uses only undeletable arcs (i.e., from $E(G) \setminus \bigcup \mathcal{B}$) and arcs of B.

Using directed flow-augmentation, we show the following.

Theorem 1.3. *Weighted Bundled Cut with Order is randomized FPT when parameterized by k and maximum size of a bundle.*

For an integer $\ell \geq 1$, the ℓ-Chain SAT problem is the Bundled Cut problem, where every bundle is a path of length at most ℓ. At ESA'13, Chitnis, Egri, and Marx [3, 5] defined ℓ-Chain SAT and showed that fixed-parameter tractability of ℓ-Chain SAT (for every fixed $\ell \geq 1$) is equivalent to the following conjecture.

Conjecture 1.4 (Conjecture 1.1 of [3]). *For every graph H, if List H-Coloring problem is polynomial-time solvable, then the vertex-deletion variant (delete at most k vertices from the input graph to obtain a yes-instance to List H-Coloring) is fixed-parameter tractable when parameterized by k.*

Clearly, a ℓ-Chain SAT instance is a Bundled Cut with Order instance with maximum bundle size at most ℓ. Hence, Theorem 1.3 implies the following.

Corollary 1.5. *ℓ-Chain SAT is (randomized) FPT when parameterized by ℓ and k, even in the weighted setting. Consequently, Conjecture 1.4 is confirmed (with randomized FPT algorithms).*

By standard reductions, a Directed Feedback Vertex Set instance can also be represented as a Bundled Cut with Order instance (with maximum size of a bundle and budget bounded linearly in the parameter of the input instance). Furthermore, in case of a weighted variant[1] the reduction preserves weights. Hence, we obtain the following.

[1] In Weighted Directed Feedback Vertex Set, the input graph is equipped with vertex weights being positive integers and one asks for a solution of cardinality at most k and total weight bounded by a threshold given on input.

Corollary 1.6. *Weighted Directed Feedback Vertex Set, parameterized by the cardinality of the deletion set, is (randomized) FPT.*

The question of parameterized complexity of Weighted Directed Feedback Vertex Set (which we answer affirmatively in Corollary 1.6) has been asked e.g. at Recent Advances in Parameterized Complexity school in December 2017 [22] and in [13].

We also take the applicability of directed flow-augmentation one step further and prove fixed-parameter tractability of Weighted Almost 2-SAT. Here, we are given a 2-CNF formula ϕ, where every clause C has weight $\omega(C)$, and positive integers k and W; the goal is to delete at most k clauses of total weight at most W to get a satisfiable instance.

Theorem 1.7. *Weighted Almost 2-SAT parameterized by k is (randomized) FPT.*

The FPT algorithm for (unweighted) Almost 2-SAT problem was one of the first applications of important separators and one of the first bridges between graph separation problems and (boolean) constraint satisfaction problems [21].

Compared to the previous applications, applying directed flow-augmentation to Weighted Almost 2-SAT in Theorem 1.7 is much more technically advanced. By known reductions [12], (Weighted) Almost 2-SAT is FPT-equivalent to (Weighted) Digraph Pair Cut. In this problem, we are given a directed graph G, a vertex $s \in V(G)$, and a number of pairs $\mathcal{T} \subseteq \binom{V(G)}{2}$. The goal is to delete at most k edges of G such that for every pair $xy \in \mathcal{T}$, either x or y is not reachable from s. (In the weighted variant, edges have positive integer weights and we also require that the total weight of deleted edges is not larger than a given threshold.)

In the process of solving a (Weighted) Digraph Pair Cut instance, an algorithm makes some decisions about some pairs $xy \in \mathcal{T}$ whether x or y will be not reachable from s in the final graph. The vertices guessed to be unreachable can be merged into one sink t; the solution Z needs to be then an st-cut, but may be required to be more than a *minimal* st-cut to satisfy other pairs of $\mathcal{T}$. On the other hand, in an inclusion-wise minimal solution Z every arc (u, v) serves some purpose: u is reachable from s in $G - Z$, but v is not. This leads to a definition of a *star st-cut*: a set $Z \subseteq E(G)$ is a star st-cut if it is an st-cut and, for every $(u, v) \in Z$, the vertex u is reachable from s in $G - Z$ but v is not. Clearly, every minimal st-cut is a star st-cut, but the reverse implication is far from true.

To apply directed flow-augmentation to Weighted Almost 2-SAT, we enhanced the procedure of Theorem 1.1 to star st-cuts: the procedure returns a set $A \subseteq V(G) \times V(G)$ and a maximum st-flow $\mathcal{P}$ in $G + A$ such that for every star st-cut Z of size at most k, with probability $2^{-O(k^4 \log k)}$ the set Z remains a star st-cut in $G + A$ and, furthermore, the edges of $Z \cap E(\mathcal{P})$ is exactly a minimum st-cut in $G + A$. That is, the arcs of A make Z contain a minimum st-cut and, furthermore, the flow $\mathcal{P}$ avoids edges of Z that are not part of the said minimum st-cut.

Future work. We believe the new technique introduced in this work opens new research directions. We would like to point out to three of them.

In the Min UNSAT(Γ) problem, we are given an instance of a Constraint Satisfaction Problem over language Γ and an integer k to delete at most k clauses to get a satisfiable instance. Here, we restrict only to binary alphabet. Note that if Γ allows unary clauses and equalities, Min UNSAT(Γ) becomes the question of minimum (undirected) st-cut and when Γ allows unary clauses and implications, Min UNSAT(Γ) becomes the question of minimum (directed) st-cut.

Recall that the undirected flow-augmentation algorithm [10] led to full classification of the Min UNSAT(Γ) for Boolean alphabet into FPT problems and $W[1]$-hard problems for languages Γ that are unable to express implication (and thus directed cuts). Up to this work, the main roadblock into generalizing this result to all Boolean languages was the ℓ-Chain SAT problem (which is easily expressible in the CSP world). Now this roadblock is removed and exploring this classification is a promising research direction.

Moreover, a similar generalization to the "star st-cut" was also present in the undirected case and was pivotal to fixed-parameter tractability of Generalized Coupled MinCut problem [10]. We believe the exemplary case of Theorem 1.7 shows that directed flow-augmentation has potential to be a handy tool in proving fixed-parameter tractability of further Boolean Min UNSAT CSP problems.

Corollary 1.6 provides an alternative algorithm for the classic Directed Feedback Vertex Set algorithm, albeit with worse time complexity. We remark that up to now only one approach to fixed-parameter tractability of Directed Feedback Vertex Set was known, namely the one using important separators [1]. A second newly opened research direction is: have we learned anything new about Directed Feedback Vertex Set? To be more precise, let us recall two other important open problems about Directed Feedback Vertex Set: is it solvable in time $2^{O(k)} n^{O(1)}$ and does it admit a polynomial kernel?

For the third direction, we mention that it is possible that directed flow-augmentation can help with resolving the question of the parameterized complexity of the Directed Multicut problem with three terminal pairs, parameterized by the solution size. Recall that this problem is FPT for two terminal pairs [6] and W[1]-hard for four [20].

2 OVERVIEW

In this section we sketch the proof of Theorem 1.1.

Note that the actual formal statement in the full version [9] is a bit more general: it handles a bit wider family of cuts than only minimal cuts (called *star st-cuts*) and provides also a handy object called *witnessing flow*. While these extensions are critical for the Weighted Almost 2-SAT algorithm, already the case of minimal st-cuts encompasses main technical ideas and thus this overview focuses only on this case. The full version of the theorem, and associated definitions, are presented at the end of this section.

Basic notation. Before we start, let us agree on basic nomenclature. All our graphs can be multigraphs. Edges may have capacities 1 or $+\infty$. The input graph has all edge capacities equal to 1. Since we never consider flows of value greater than k, a $+\infty$-capacity edge is equivalent to $(k+1)$ copies of an edge of capacity 1. If G is a graph and $A \subseteq V(G) \times V(G)$, then $G + A$ is the graph G with every arc of A added *with capacity* $+\infty$.

For vertices $s, t \in V(G)$, an *st-flow* is a collection $\mathcal{P}$ of paths from s to t such that no edge of capacity 1 lies on more than one path of $\mathcal{P}$. The *value* of the flow is the number of paths. By $\lambda_G(s,t)$ we denote the maximum possible value of an st-flow; note that it may happen that $\lambda_G(s,t) = +\infty$. An st-flow is a *maximum st-flow* or *st-maxflow* if its value is $\lambda_G(s,t)$. By convention, if $\lambda_G(s,t) = +\infty$, then any st-flow that contains a path with all edges of capacity $+\infty$ is considered an st-maxflow.

A set $Z \subseteq E(G)$ is an *st-cut* if it contains no edge of capacity $+\infty$ and there is no path from s to t in $G - Z$. An st-cut Z is *minimal* if no proper subset of Z is an st-cut and *minimum* (or *st-mincut*) if it has minimum possible cardinality. By Menger's theorem, if $\lambda_G(s,t) < +\infty$ then the size of every st-mincut is exactly $\lambda_G(s,t)$ and there are no st-cuts if $\lambda_G(s,t) = +\infty$.

For an st-cut Z, the *s-side* of Z is the set of vertices reachable from s in $G - Z$, and the *t-side* of Z is the complement of the s-side. Note that this is not symmetric as we do not mandate that t is reachable from all elements of the t-side in $G - Z$. If we have two (inclusion-wise) minimal st-cuts C_1 and C_2 such that both endpoints of every edge of C_1 lie in the s-side of C_2 (and thus both endpoints of every edge of C_2 lie in the t-side of C_1), then a vertex is *between* C_1 and C_2 if it is both in the t-side of C_1 and s-side of C_2.

One flow-augmentation step. If we do not pay particular attention to the exact bound on the probability of a success and we are happy with any $1/f(k)$ bound for a computable function f, it suffices to prove the following:

Theorem 2.1. *There exists a computable function $f : \mathbb{N} \to \mathbb{Z}_+$ and a polynomial-time algorithm that, given a directed graph G, two vertices $s, t \in V(G)$, and an integer k, outputs a set $A \subseteq V(G) \times V(G)$ such that $\lambda_{G+A}(s,t) > \lambda_G(s,t)$ and for every minimal st-cut Z of size at most k that is* not *a* minimum *st-cut, Z remains an st-cut in $G + A$ with probability at least $1/f(k)$.*

Indeed, an algorithm for Theorem 1.1 may first randomly guess the size of the minimal st-cut Z it cares about and then repeat the algorithm of Theorem 2.1, adding the output set A to the graph G, as long as $\lambda_G(s,t)$ is smaller than the guessed size of Z. Thus, in the remainder of this section we sketch the proof of Theorem 2.1.

For the proof of Theorem 2.1, we fix one minimal st-cut Z of size at most k that is not a minimum st-cut and we bound from below the probability that the algorithm never adds an arc from the s-side to the t-side of Z.

The algorithm for Theorem 2.1 is recursive. We start with computing an st-maxflow $\mathcal{P}$ in G; the goal is to push somehow an extra unit of flow from s to t by adding arcs so that no added arc violates the cut Z. The algorithm tries to isolate smaller and smaller parts of the graph through which it tries to push an extra unit of flow, guessing some properties of the cut Z along the way to guide the recursion. To obtain the desired success probability, we need to very restrictively guess properties of the cut Z.

Natural candidates as separations between distinct areas for recursive calls are minimum st-cuts. This was also the case for the undirected counterpart of [10]: the algorithm there constructed a maximal sequence of noncrossing and pairwise disjoint minimum st-cuts, partitioned them into minimal subsegments where the graph in-between was connected, and recursed on them.

In the undirected case, the edges of a minimum st-cut C is the entire "interface" between the s-side and t-side of C. In the directed case, there can be an unbounded number of arcs with tails in the t-side of C and heads in the s-side of C. This makes the interaction between the s-side of C and t-side of C much more complicated and harder to grasp in recursive calls.

One of the main ideas of the proof of Theorem 2.1 — and one of the main differences between it and its undirected counterpart [10] — is a very careful choice of the minimum st-cuts we want to separate along. Let us now introduce it in detail.

Reachability patterns, leaders, and mincut sequences. An *instance* is a tuple $\mathcal{I} = (G, s, t, k)$ where G is a directed graph, $s, t \in V(G)$, and k is a nonnegative integer. An *instance with a flow* is a pair $(\mathcal{I}, \mathcal{P})$ where $\mathcal{I} = (G, s, t, k)$ is an instance and $\mathcal{P} = \{P_1, P_2, \ldots, P_\lambda\}$ is an st-flow. In an *instance with a maximum flow* we additionally require $\lambda = \lambda_G(s,t)$, that is, $\mathcal{P}$ is a maximum st-flow. With an instance with a flow $(\mathcal{I}, \mathcal{P})$ we associate the residual graph $G^{\mathcal{P}}$.

For an instance with a maximum flow $(\mathcal{I} = (G, s, t, k), \mathcal{P} = \{P_1, \ldots, P_\lambda\})$, a *reachability pattern* is a directed graph H with vertex set $V(H) = [\lambda]$, that is, each vertex $i \in V(H)$ corresponds to a path $P_i \in \mathcal{P}$, that contains a self-loop at every vertex. The *pattern associated with* $(\mathcal{I}, \mathcal{P})$ is a graph H with $V(H) = [\lambda]$ and $(i,j) \in E(H)$ if and only if there exists $v \in V(P_i) \setminus \{s,t\}$ and $u \in V(P_j) \setminus \{s,t\}$ such that there is a v to u path in $G^{\mathcal{P}}$. Note that a pattern associated with $(\mathcal{I}, \mathcal{P})$ is a reachability pattern if and only if every path P_i has at least one internal vertex.

For a vertex $v \in V(G)$, the set $\mathrm{RReach}(v)$ is the set of vertices reachable from v in $G^{\mathcal{P}}$. For a vertex $v \in V(G)$ and an index $i \in [\lambda]$, the *last vertex on P_i residually reachable from v*, denoted $\mathrm{LastReach}(v, P_i)$, is the last (closest to t) vertex u on P_i that is reachable from v in $G^{\mathcal{P}}$. Note that if u is reachable from v in $G^{\mathcal{P}}$, then all vertices preceding u on P_i are also reachable from v in $G^{\mathcal{P}}$, that is, the set of vertices of P_i reachable from v in $G^{\mathcal{P}}$ form a prefix of P_i. Furthermore, note that if v_1 and v_2 are on P_j and v_1 is earlier than v_2 on P_j, then $\mathrm{LastReach}(v_1, P_i)$ is not later than $\mathrm{LastReach}(v_2, P_i)$ on P_i (they may be equal).

Let C be an st-mincut in $\mathcal{I}$ and let H be a reachability pattern. For $i \in [\lambda]$, the *leader of the path P_i after C* (with respect to the pattern H), denoted $\mathrm{leader}_H(C, i)$, is the first (closest to s) vertex v on P_i such that for every $(i,j) \in E(H)$ the vertex $\mathrm{LastReach}(v, P_j)$ is after C on P_j. A few remarks are in place. First, the notion of the leader is well-defined for any H as the vertex t is always a feasible candidate. Second, since a reachability pattern is required to contain a self-loop at every vertex, for every $i \in [\lambda]$ there is at least one edge $(i,j) \in E(H)$ to consider. Third, as C is oriented from the t-side to the s-side in $G^{\mathcal{P}}$, $\mathrm{leader}_H(C, i)$ is after C on P_i and, furthermore, the entire path in $G^{\mathcal{P}}$ from $\mathrm{leader}_H(C, i)$ to $\mathrm{LastReach}(v, P_j)$ for every $(i,j) \in E(H)$ lies on the t-side of C.

For an st-mincut C and a reachability pattern H, we define the *mincut H-subsequent to C* as follows. If $t \in \mathrm{RReach}(\mathrm{leader}_H(C, i))$ for some $i \in [\lambda]$, the mincut H-subsequent to C is undefined. Otherwise, one can argue that $\delta^+(\mathrm{RReach}(\mathrm{leader}_H(C, i)))$ is an st-mincut for every $i \in [\lambda]$. Let $X = \bigcup_{i \in [\lambda]} \mathrm{RReach}(\mathrm{leader}_H(C, i))$. By submodularity, $C' := \delta^+(X)$ is also an st-mincut; we proclaim C' to be the mincut H-subsequent to C. Observe that, by definition, for every $i \in [\lambda]$, the leader $\mathrm{leader}_H(C, i)$ lies in the t-side of C and s-side of

C', the set $\mathrm{RReach}(\mathrm{leader}_H(C,i))$ lies in the s-side of C' and for every $(i,j) \in E(H)$ any path in $G^{\mathcal{P}}$ from $\mathrm{leader}_H(C,i)$ to a vertex on P_j in the t-side of C (in particular, to $\mathrm{LastReach}(\mathrm{leader}_H(C,i), P_j)$) lies entirely in the t-side of C and s-side of C'. Furthermore, for every tail of an edge $(u,v) \in C' \cap E(P_j)$, either u is the leader $\mathrm{leader}_H(C,j)$ or there exists $\mathrm{leader}_H(C,i)$ (possibly with $i = j$) such that $u \in \mathrm{RReach}(\mathrm{leader}_H(C,i))$. Note that in the latter case, there is no st-mincut with $\mathrm{leader}_H(C,i)$ in its s-side and u in its t-side because there is a residual path from $\mathrm{leader}_H(C,i)$ to u. This proves the following observation.

Lemma 2.2. *Let $((G,s,t,k),\mathcal{P})$ be an instance with maximum flow, let H be the reachability pattern associated with it, let C be an st-mincut and let C' be an st-mincut whose endpoints are entirely in the t-side of C. Let G' be a graph constructed from G by contracting the s-side of C onto s and t-side of C' onto t and let $\mathcal{P}'$ be an st-flow in G' consisting of the subpaths of paths of $\mathcal{P}$ between the edge of C and the edge of C'. Suppose $\mathcal{P}'$ is an st-maxflow in G'. If C' is H-subsequent to C, then the pattern associated with $((G',s,t,k),\mathcal{P}')$ is still H. Moreover, the st-mincut H-subsequent to C is an st-mincut closest to C with this property.*

For a reachability pattern H, an *H-sequence of mincuts* is a sequence $C_1, C_2, \ldots, C_\ell$ of mincuts defined as follows. C_1 is the st-mincut closest to s and for $a > 1$ the mincut C_a is the mincut H-subsequent to C_{a-1}, as long as it is defined. Observe that for every $1 \le a < b \le \ell$ and $i \in [\lambda]$ we have that the edge of C_a on P_i lies strictly before the edge of C_b on P_i. That is, the s-side of C_a is contained in the s-side of C_b and, furthermore, on every path P_i the s-side of C_a is a strict subset of the s-side of C_b.

Preprocessing and the base case of the recursion. The input to a recursive call is an instance with a flow $(\mathcal{I} = (G,s,t,k), \mathcal{P} = \{P_1, \ldots, P_\lambda\})$ and the goal is to return a set $A \subseteq V(G) \times V(G)$ such that $\lambda < \lambda_{G+A}(s,t)$ and for every minimal st-cut Z with $\lambda < |Z| \le k$ the set Z remains an st-cut in $G + A$ with good enough probability. In the initial, root call we set $\mathcal{P}$ to be any st-maxflow.

A recursive call first performs a few preprocessing steps. If $\lambda < \lambda_G(s,t)$, it is safe to return $A = \emptyset$. If $\lambda_G(s,t) = 0$ or $\lambda_G(s,t) \ge k$, then there is no minimal st-cut of size at most k that is not a minimum st-cut. If this is the case, we can return $A = \{(s,t)\}$. Henceforth we assume $0 < \lambda_G(s,t) < k$ and that $\mathcal{P}$ is an st-maxflow.

A relatively standard preprocessing step (that thus we do not describe here) can ensure that both $\delta^+(s)$ and $\delta^-(t)$ (the set of edges with tail in s and the set of edges with head in t, respectively) are disjoint and are both st-mincuts. In particular, this implies that the pattern H associated with $\mathcal{I}$ and $\mathcal{P}$ is a reachability pattern.

In the recursive steps, we will aim at either decreasing $2k - \lambda$ or keeping k and λ intact, but decreasing the number of edges in the pattern H associated with $(\mathcal{I}, \mathcal{P})$. Intuitively, less edges in $E(H)$ means that the graph G is somewhat simpler.

In the base case of the recursion, H consists of λ isolated vertices, with a loop on every vertex of $V(H) = [\lambda]$. In other words, $G^{\mathcal{P}}$ contains no path from a vertex of $V(P_i) \setminus \{s,t\}$ to a vertex of $V(P_j) \setminus \{s,t\}$ for any distinct $i,j \in [\lambda]$, or equivalently $G \setminus \{s,t\}$ contains no path from a vertex of $V(P_i)$ to a vertex of $V(P_j)$. In this case, the paths P_i are somewhat independent in G and it is relatively easy to show the following. (Here, an edge $e \in E(G)$ is *bottleneck* if there exists a minimum st-cut C with $e \in C$.)

Lemma 2.3. *Let $(\mathcal{I}, \mathcal{P})$ be an instance with maximum flow with proper boundaries and let H be its reachability pattern. If $|V(H)| = |E(H)|$, then for every minimal st-cut Z that is not a minimum cut, there exists $i \in [\lambda]$ such that no edge of $Z \cap E(P_i)$ is a bottleneck edge.*

Lemma 2.3 allows the following step: sample $i \in [\lambda]$ at random and return A consisting of duplicates of all bottleneck edges of P_i. Lemm 2.3 ensures that with probability at least $\lambda^{-1} \ge k^{-1}$, a minimal st-cut that is not a minimum st-cut remains an st-cut in $G + A$, while $\lambda_{G+A}(s,t) > \lambda_G(s,t)$ follows from the fact that we duplicated all bottleneck edges on one flow path.

If $|E(H)| > |V(H)|$, we aim at decomposing the graph further and make some recursive calls. We compute the H-sequence of mincuts $C_1, C_2, \ldots, C_\ell$. Note that we have $C_1 = \delta^+(s)$ and it is easy to see that C_2 is defined, and thus $\ell \ge 2$. We set a threshold ℓ^{big} depending on k (in the actual proof we have ℓ^{big} being polynomially bounded in k) and split into two cases: $\ell \le \ell^{\mathrm{big}}$ or $\ell > \ell^{\mathrm{big}}$.

Small ℓ case. In this case, we can afford guessing (by random coin flip) for each $i \in [\ell]$ and $e \in C_i$, whether the endpoints of $e = (u,v)$ are in the s-side or t-side of Z.

There are a few simple cases. If for some $e = (u,v)$ we guessed that u is in the s-side of Z and v is in the t-side, then $e \in Z$; we can recurse on $G' := G - e$ and $\mathcal{P}' := \mathcal{P} \setminus \{P\}$ with parameter $k - 1$, where $P \in \mathcal{P}$ is a flow path containing e, and obtain a set A', and return $A := A' \cup \{(s,u),(v,t)\}$. If for some $e = (u,v)$ we guessed that u is in the t-side of Z and v is in the s-side of Z, then we can return $A = \{(s,v),(u,t)\}$ as $s - v - u - t$ is an augmenting path in $G + A$ with respect to $\mathcal{P}$, so $\lambda_{G+A}(s,t) > \lambda$. Finally, if for every $e \in C_\ell$, both endpoints of e are in the s-side of Z, then we can recurse on G' being the graph G with the s-side of C_ℓ contracted onto s; the fact that $C_{\ell+1}$ is undefined implies that in the recursive call the associated pattern is a proper subgraph of H.

In the main case, let A_0 consist of edges (s,v) for any endpoint v of an edge $e \in C_i$, $i \in [\ell]$ that is guessed to be in the s-side of Z and edges (u,t) for any endpoint u of an edge $e \in C_i$, $i \in [\ell]$ that is guessed to be in the t-side of Z. If the guess is correct, Z remains an st-cut in $G + A_0$. If $\lambda_{G+A_0}(s,t) > \lambda$, then we can return A_0, so assume otherwise.

Let C be the minimum st-cut in $G + A_0$ that is closest to t. For every endpoint of an edge of C, we guess whether it is in the s-side or t-side of Z. We again have a few simple cases. If for some $e = (u,v) \in C$, u is in the s-side of Z and v is in the t-side of Z, we have $e \in Z$ and we recurse on $G - e$ as before. If a head v of an edge of C is in the s-side of Z, then we return $A := A_0 \cup \{(s,v)\}$; $\lambda_{G+A}(s,t) > \lambda$ follows from the fact that C is the closest to t minimum st-cut in $G + A_0$.

We are left with the most interesting case where C is completely in the t-side of Z. We define $a \in [\ell]$ to be the maximum index such that all endpoints of C_a are in the s-side of Z. This is well-defined as $C_1 = \delta^+(s)$ is in the s-side of Z (as we are not in any of the simple cases). Also we have $a < \ell$ and C_{a+1} is defined (again, we are not in any of the simple cases) and also C lies entirely in the t-side of C_a. Let G' be the graph G with the s-side of C_a contracted onto s and the t-side of C contracted onto t. Let $\mathcal{P}'$ be the flow $\mathcal{P}$ projected onto G' (i.e., we shorten all paths to start from the edges of C_a and end with the edges of C).

The crucial observation is that the pattern associated with G' and $\mathcal{P}'$ is a *proper* subgraph of H. Indeed, by the choice of a (and exclusion of the simple cases), there is at least one tail u of an edge of C_{a+1} that is in the t-side of Z; this vertex u lies in the t-side of C_a and the s-side of C_{a+1}. Moreover, u is in the t-side of C as the edge (u,t) has been added to A_0 and C is an st-mincut in $G + A_0$. Since C_{a+1} is the mincut H-subsequent to C_a, by Lemma 2.2 the subgraph between C_a and C_{a+1} maintains all reachability (in the residual graph) described by H, but C_{a+1} is chosen to be the closest-to-C_a minimum st-cut that satisfies Lemma 2.2. Since u is in the t-side of C but s-side of C_{a+1}, we can infer that for at least one edge of H, the corresponding reachability is not present in G' and $\mathcal{P}'$. We recurse on G' and $\mathcal{P}'$'; this decrease in $|E(H)|$ is a progress in the recursive step.

We remark here that the above progress in the recursion is the main reason to introduce the notions of reachability patterns and sequences of mincuts. It seems to us very delicate; we were not able to reproduce a similar measure of progress with different ways of defining sequences of cuts $C_1, \ldots, C_\ell$.

Large ℓ case. The progress in the previous case is possible partially due to the fact that ℓ is bounded in k and we could afford guessing the s-side/t-side assignment of all endpoints of all cuts C_i. If ℓ is large (unbounded in k), we need to resort to a color-coding step to split the long sequence of C_i's into smaller chunks that are eligible for the small ℓ case.

Let Z_{ts} be the set of edges of paths of $\mathcal{P}$ whose tail is in the t-side of Z but head is in the s-side of Z. On each flow path P_i, we have $|E(P_i) \cap Z| = |E(P_i) \cap Z_{ts}| + 1$ and thus $|Z_{ts}| \leq k - \lambda$. An index $a \in [\ell]$ is *touched* if there is an endpoint of an arc of $Z \cup Z_{ts}$ that is in the t-side of C_a and s-side of C_{a+1}. We have at most $2|Z \cup Z_{ts}| \leq 4k - 2\lambda$ touched indices.

Recall that the construction of the cuts $C_1, C_2, \ldots, C_\ell$ ensures that, for every $a \in [\ell - 1]$, between C_a and C_{a+1} one can find a path from P_i to P_j in the residual graph for every $(i,j) \in E(H)$. On the other hand, $G - \{s,t\}$ features no path in the residual graph from P_i to P_j if $(i,j) \notin E(H)$. In some sense, it means that the connectivity between paths P_i that is present in the entire graph G is repeatedly realized between each two consecutive cuts C_a.

The fact that we are operating with the residual graph was essential for the small ℓ case. Here, we need to depart from the residual graph and observe the following: for every $(i,j) \in E(H)$ and $1 \leq a \leq \ell - \lambda$, there is a path in G from P_i to P_j that starts between C_a and C_{a+1}, ends between $C_{a+\lambda-1}$ and $C_{a+\lambda}$, and is contained between C_a and $C_{a+\lambda}$.

This has a number of consequences. First, if ℓ^{big} is large enough, then H must be transitive.

Second, for every $a \in [\ell]$ let $L_a \subseteq [\lambda]$ be the set of indices i such that both endpoints of the unique edge of $E(P_i) \cap C_a$ are in the s-side of Z. Observe that if for some $1 \leq a \leq \ell - \lambda$, all indices $a \leq b \leq a + \lambda$ are untouched, then the set $L_{a+\lambda}$ is *downward-closed*: there is no arc $(i,j) \in E(H)$ such that $i \in L_{a+\lambda}$ but $j \notin L_{a+\lambda}$.

Third, observe that if $L \subseteq [\lambda]$ is downward-closed, then for every $i \in L$ and $j \notin L$, the graph $G - \{s,t\}$ contains no path from P_i to P_j. Hence, if we denote by $\mathrm{cl}(L_a)$ be the minimal superset of L_a that is downward-closed, we have that $L_b \subseteq \mathrm{cl}(L_a)$ whenever $1 \leq a \leq b \leq \ell$.

In particular, if L_a is downward-closed, we have the following.

1. $L_b \subseteq L_a$ for $b \geq a$.
2. For every $i \in L_a$, the entire prefix of P_i up to the edge of C_a is in the s-side of Z, as every path from a vertex v on such prefix to t needs to intersect $C_a \cap \bigcup_{j \in L_a} E(P_j)$ (and Z is a minimal st-cut).
3. Symmetrically for every $i \notin L_a$, the entire suffix of P_i from the edge of C_a is in the t-side of Z.
4. There is no edge $e = (u,v) \in Z$ with u in the t-side of C_a and v in the s-side of C_a. Indeed, if $e = (u,v)$ were such an edge, then by minimality of Z the graph $G - Z$ would need to feature a path from s to v and from u to t; the first path needs to traverse an edge of $C_a \cap E(P_i)$ for $i \in L_a$ and the second path needs to traverse an edge of $C_a \cap E(P_j)$ for $j \notin L_a$, hence in between we have a path from P_i to P_j in $G - \{s,t\}$.

Let T be the set of touched indices. We say that two indices $a < b$ are *close* if $b - a \leq \lambda$. The set T partitions into *blocks*: maximal sets of indices such that two consecutive indices are close to each other. Let $B_1, \ldots, B_r$ be the blocks and for $1 \leq \alpha \leq r$, let a_α and b_α be the minimum and maximum index of the block B_α. We have $b_\alpha - a_\alpha = O(k\lambda)$ while indices $a_\alpha - \lambda, a_\alpha - \lambda + 1, \ldots, a_\alpha - 1$ and $b_\alpha + 1, \ldots, b_\alpha + \lambda$ are untouched (we ignore here boundary cases when $a_\alpha \leq \lambda$ or $b_\alpha + \lambda > \ell$ which are easy to adjust to). In particular, $L_{a_\alpha - 1}$ and $L_{b_\alpha + \lambda}$ are downward-closed. Denote $L_\alpha^{\leftarrow} = L_{a_\alpha - 1}$, $L_\alpha^{\rightarrow} = L_{b_\alpha + \lambda}$, and $D^\alpha = L_\alpha^{\leftarrow} \setminus L_\alpha^{\rightarrow}$.

An important observation from the downward-closedness of $L_\alpha^{\leftarrow}$ and $L_\alpha^{\rightarrow}$ is the following: for every edge $e \in Z$, there is a block B_α such that *both* endpoints of e lie between C_{a_α} and $C_{b_\alpha+1}$. Hence, Z partitions into $Z_1 \uplus Z_2 \uplus \ldots \uplus Z_r$ where Z_α is the set of edges of Z with both endpoints between C_{a_α} and $C_{b_\alpha+1}$.

For every $\alpha \in [r]$, define a graph G_α as follows: contract the s-side of $C_{a_\alpha - 1}$ onto s, contract the t-side of $C_{b_\alpha + \lambda}$ onto t, and from the edges incident with s or t leave only the ones on paths P_i for $i \in D_\alpha$. The flow paths P_i for $i \in D_\alpha$ naturally project to a flow $\mathcal{P}_\alpha$ of size $|D^\alpha|$ in G_α. The important observation again from the downward-closedness of $L_\alpha^{\leftarrow}$ and $L_\alpha^{\rightarrow}$ is as follows: in G_α, Z_α is a minimal st-cut.

This in particular implies that $D^\alpha \neq \emptyset$ and thus

$$[\lambda] = L_1^{\leftarrow} \supsetneq L_1^{\rightarrow} = L_2^{\leftarrow} \supsetneq L_2^{\rightarrow} = \ldots = L_r^{\leftarrow} \supsetneq L_r^{\rightarrow} = \emptyset.$$

We would like to recurse on instances $\mathcal{I}_\alpha = (G_\alpha, \mathcal{P}_\alpha, k_\alpha)$ where $k_\alpha = |Z_\alpha|$, but there is a significant problem: we do not know the blocks B_α. However, following a similar step in the undirected algorithm of [10], we can deal with it with color-coding.

Sample $\Gamma \subseteq [\ell]$, aiming at follows: for every a that is close to a touched index, we want $a \in \Gamma$ if and only if a is touched. If we put every $a \in [\ell]$ into Γ with probability 0.5 independently of other indices, the success probability is $2^{-O(k\lambda)}$. Then, the indices of Γ partition into Γ-blocks: one Γ-block is a maximal subset of Γ in which two consecutive indices are close. If the guess is successful, every block is a Γ-block, but there may be numerous Γ-blocks that are in fact wholy untouched.

We can guess $r \leq 4k - 2\lambda$, sets $L_\alpha^{\leftarrow}, L_\alpha^{\rightarrow}$ as well as sizes $k_\alpha = |Z_\alpha|$ for $\alpha \in [r]$. For every Γ-block, we guess its "type" in $[r]$, aiming that a true block B_α has type α. For a Γ-block B of type $\alpha(B)$, we construct from it an instance $(G_B, s, t, k_{\alpha(B)})$ with flow $\mathcal{P}_B$ in the same way as we constructed $G_{\alpha(B)}$ and $\mathcal{P}_{\alpha(B)}$ for $B_{\alpha(B)}$. If $k_{\alpha(B)} > |D_{\alpha(B)}|$ (i.e., $Z_{\alpha(B)}$ is not a minimum cut in $G_{\alpha(B)}$) we recurse on it, obtaining a set A_B (we denote $A_B = \emptyset$ for cases $k_{\alpha(B)} = D_{\alpha(B)}$).

The final set A consists of:

(1) for every Γ-block B with minimum index a and maximum index b, the following edges:
   (a) for every $(u,v) \in A_B$:
      - if $u \neq s$ and $v \neq t$, just the arc (u,v) itself;
      - if $u = s$ and $v \neq t$, all arcs (u', v) where u' ranges over all tails of edges of C_{a-1} on paths P_i for $i \in D_\alpha$;
      - if $u \neq s$ and $v = t$, all arcs (u, v') where v' ranges over all tails of edges of $C_{b+\lambda}$ on paths P_i for $i \in D_\alpha$;
      - if $u = s$ and $v = t$, all arcs (u', v') where u' ranges over all tails of edges of C_{a-1} on paths P_i for $i \in D_\alpha$ and v' ranges over all tails of edges of $C_{b+\lambda}$ on paths P_i for $i \in D_\alpha$;
   (b) for every $i \in [\lambda] \setminus D_\alpha$ and arc from the tail of the edge of $C_{a-1} \cap E(P_i)$ to the tail of the edge of $C_{b+\lambda} \cap E(P_i)$;
(2) for every $1 \leq c \leq \ell$ for which there is no Γ-block B with minimum index a and maximum index b and $a-1 \leq c \leq b+\lambda$:
   - for every $i \in [\lambda]$, an edge from the tail of the edge of $C_c \cap E(P_i)$ to the tail of the edge of $C_{c+1} \cap E(P_i)$;
(3) for every $1 \leq c \leq \ell$ for which there is no Γ-block B with minimum index a and maximum index b and $a-1 < c < b+\lambda$:
   - for every $1 \leq \alpha \leq r$, for every $i, j \in D_\alpha$, an edge from the tail of the edge of $C_c \cap E(P_i)$ to the tail of the edge of $C_c \cap E(P_j)$.

To see that $\lambda_{G+A}(s,t) > \lambda$, notice first that as Z is not a minimum st-cut, there exists an index α where $k_\alpha = |Z_\alpha| > D_\alpha$. We push an extra unit of flow along flow paths P_i, $i \in D_\alpha$ as follows.

- For every Γ-block B of type α (with minimum index a and maximum index b), we push $|D_\alpha| + 1$ units of flow from the tails of edges of C_{a-1} on paths P_i for $i \in D_\alpha$ to the tails of edges of $C_{b+\lambda}$ on paths P_i for $i \in D_\alpha$ using edges of A_B (Point 1a) and inductive assumption from the recusive call.
- For every Γ-block B of type $\alpha' \neq \alpha$, we push $|D_\alpha| + 1$ units of flow along edges added in Point 1b for paths P_i, $i \in D_\alpha$.
- For every $1 \leq c \leq \ell$ for which there is no Γ-block B with minimum index a and maximum index b and $a-1 \leq c \leq b+\lambda$, we push $|D_\alpha| + 1$ units of flow along edges added in Point 2 for paths P_i, $i \in D_\alpha$.
- We use edges added in Point 3 to reshuffle the flow on paths P_i, $i \in D_\alpha$ between the steps above, if needed.

Let us now briefly analyse the progress in the above recursion. If $r > 1$, then all recursive calls treat strictly smaller value of k. For $r = 1$, we have $D_1 = \lambda$ and possibly $k_\alpha = k$. However, then the recursive calls fall into the "small ℓ case" as the cuts C_c inside recursive call remain an H-sequence of mincuts. This is the desired progress and it finishes the overview of the proof of Theorem 2.1.

2.1 Star cuts

For more advanced applications of the framework, such as the algorithm for Weighted Almost 2-SAT, we need the full version of Theorem 2.1, which we present here. The proofs are deferred to the full version [9].

An st-cut Z is a *star st-cut* if for every $(u,v) \in Z$, in the graph $G - Z$ there is a path from s to u but there is no path from s to v. Note that every minimal st-cut is a star st-cut, but the implication in the other direction does not hold in general. For a star st-cut Z in G, by $\mathrm{core}_G(Z) \subseteq Z$ we denote the set of arcs $(u,v) \in Z$ such that there exists a path from v to t in $G - Z$. We drop the subscript if the graph G is clear from the context. We observe the following.

Lemma 2.4. *If Z is a star st-cut in a graph G, then $\mathrm{core}(Z)$ is a minimal st-cut.*

We say that a set of arcs $A \subseteq V(G) \times V(G)$ is *compatible* with a star st-cut Z if the following holds: for every $v \in V(G)$, there is a path from s to v in $G - Z$ if and only if there is a path from s to v in $(G + A) - Z$. Equivalently: the s-sides and t-sides of Z are equal in G and $G + A$, or no arc of A has its tail in the s-side of Z in G and its head in the t-side of Z in G.

An immediate yet important observation is as follows.

Lemma 2.5. *If Z is a star st-cut in G and $A \subseteq V(G) \times V(G)$ is compatible with Z, then Z is a star st-cut in $G + A$ as well.*

We remark that albeit in the setting of Lemma 2.5 the cut Z remains a star st-cut in $G + A$, it may happen that $\mathrm{core}_G(Z) \subsetneq \mathrm{core}_{G+A}(Z)$, as the arcs of A may add some new reachability towards t.

Assume Z is a star st-cut in G such that $\mathrm{core}(Z)$ is an st-mincut. An st-maxflow $\mathcal{P}$ is a *witnessing flow* if $E(\mathcal{P}) \cap Z = \mathrm{core}(Z)$, that is, $\mathcal{P}$ contains one edge of $\mathrm{core}(Z)$ on each flow path and no other edge of Z. A witnessing flow may not exist in general, even if $\mathrm{core}(Z)$ is an st-mincut. In contrast, on undirected graphs an arbitrary st-maxflow is a witnessing flow for the counterpart of $\mathrm{core}(Z)$ when the latter is an st-mincut [10]. However, our flow-augmentation procedure will ensure that not only $\mathrm{core}(Z)$ becomes an st-mincut in the augmented graph, but also a flow is returned that is a witnessing flow in the augmented graph. Formally, for a star st-cut Z in G, $A \subseteq V(G) \times V(G)$, and an st-maxflow $\widehat{\mathcal{P}}$ in $G + A$, we say that $(A, \widehat{\mathcal{P}})$ is *compatible* with Z if A is compatible with Z, $\mathrm{core}_{G+A}(Z)$ is an st-mincut in $G + A$, and $\widehat{\mathcal{P}}$ is a witnessing flow for Z in $G + A$. Our flow-augmenting procedure will return a pair $(A, \widehat{\mathcal{P}})$ that is compatible with a fixed star st-cut Z with good probability.

Theorem 2.6. *There exists a polynomial-time algorithm that, given an instance $\mathcal{I} = (G, s, t, k)$, returns a set $A \subseteq V(G) \times V(G)$ and an st-maxflow $\widehat{\mathcal{P}}$ in $G + A$ such that for every star st-cut Z of size at most k, with probability $2^{-O(k^4 \log k)}$ the pair $(A, \widehat{\mathcal{P}})$ is compatible with Z.*

3 APPLICATIONS: WEIGHTED st-CUT, WEIGHTED DFVS, WEIGHTED CHAIN SAT

3.1 Tractable case of Weighted Bundled Cut

An instance of Bundled Cut consists of a directed graph G, vertices $s, t \in V(G)$, a nonnegative integer k, and a family $\mathcal{B}$ of pairwise disjoint subsets of $E(G)$. An element $B \in \mathcal{B}$ is called a *bundle*. An edge that is part of a bundle is *deletable*, otherwise it is *undeletable*. A *cut* in a Bundled Cut instance $\mathcal{I} = (G, s, t, k, \mathcal{B})$ is an st-cut Z that does not contain any undeletable edge, that is, $Z \subseteq \bigcup \mathcal{B}$. A cut Z *touches* a bundle $B \in \mathcal{B}$ if $Z \cap B \neq \emptyset$. The *cost* of a cut Z is the number of bundles it touches. A cut Z is a *solution* if its cost is at most k. The Bundled Cut problem asks if there exists a solution to the input instance.

An instance of Weighted Bundled Cut consists of a Bundled Cut instance $\mathcal{I} = (G, s, t, k, \mathcal{B})$ and additionally a weight function $\omega : \mathcal{B} \to \mathbb{Z}_+$ and an integer $W \in \mathbb{Z}_+$. The *weight* of a cut Z in $\mathcal{I}$ is the total weight of all bundles it touches. A solution Z to $\mathcal{I}$ is a solution to the Weighted Bundled Cut instance $(\mathcal{I}, \omega, W)$ if additionally the weight of Z is at most W. The Weighted Bundled Cut problem asks if there is a solution to the input instance. Note that any Bundled Cut instance can be treated as a Weighted Bundled Cut instance by setting ω uniformly equal 1 and $W = k$.

When parameterized by k, it is well known that Bundled Cut is W[1]-hard even if all bundles are of size 2 [18]. To get tractability, we define the following restriction called *Weighted Bundled Cut with Order* where we require the following: in the input instance $((G, s, t, k, \mathcal{B}), \omega, W)$, for every $B \in \mathcal{B}$ one can order the arcs of B as $e_1^B, \ldots, e_{|B|}^B$ such that for every $1 \le i < j \le |B|$ there exists a path $P_{i,j}^B$ in G that goes from one endpoint of e_i^B to one endpoint of e_j^B and uses only undeletable arcs and arcs of B.

In this subsection we prove the following theorem.

Theorem 3.1. *Weighted Bundled Cut with Order is randomized FPT when parameterized by k and maximum size of a bundle.*

The main workhorses behind the proof of Theorem 3.1 are the flow-augmentation routine and the following lemma.

Lemma 3.2. *Assume we are given a Weighted Bundled Cut with Order instance $\mathcal{I} = ((G, s, t, k, \mathcal{B}), \omega, W)$ Then one can in time FPT with parameters $\lambda_G(s,t)$, maximum size of a bundle, and k check if there is a solution to $\mathcal{I}$ that is an st-mincut, where for $\lambda_G(s,t)$ we treat every deletable edge as of capacity 1 and every undeletable edge as of capacity $+\infty$.*

Proof of Theorem 3.1 using Lemma 3.2. Let b be the maximum size of a bundle. We will describe a randomized algorithm that runs in time FPT in b and k and, if given a yes-instance, answers yes with probability $2^{-O((bk)^4 \log(bk))}$, and never answers yes to a no-instance. Repeating the algorithm $2^{O((bk)^4 \log(bk))}$ times gives the desired algorithm.

Consider a solution Z to the input Weighted Bundled Cut with Order instance $\mathcal{I} = ((G, s, t, k, \mathcal{B}), \omega, W)$. Without loss of generality, we can assume that Z is a minimal st-cut. Note that $|Z| \le bk$. Apply flow-augmentation (Theorem 1.1) to G and parameter bk, obtaining a set A, and add all arcs of A to G as undeletable arcs. Note that adding arcs A to G does not break the requirements for a Weighted Bundled Cut with Order instance. With probability $2^{-O((bk)^4 \log(bk))}$ the set Z is still a solution in the final instance and is an st-mincut. Run the algorithm of Lemma 3.2, returning its answer. □

Thus, we are left with proving Lemma 3.2.

Proof of Lemma 3.2. For every bundle $B \in \mathcal{B}$, fix an order $B = (e_1^B, \ldots, e_{|B|}^B)$ as in the definition of a Weighted Bundled Cut with Order instance. Let b be the maximum size of a bundle.

Compute a maximum st-flow $\mathcal{P} = \{P_1, P_2, \ldots, P_\lambda\}$ where $\lambda := \lambda_G(s,t)$. Fix a hypothetical solution Z; note that Z contains exactly one deletable edge on every path P_i; denote by f_i the unique edge of $E(P_i) \cap Z$. Branch, guessing the following information about Z. First, guess the number $\kappa \le k$ of bundles touched by Z. Let those bundles be $F_1, F_2, \ldots, F_\kappa \in \mathcal{B}$. For every $i \in [\lambda]$, guess the indices $\alpha(i) \in [\kappa]$ and $\beta(i) \in [b]$ such that $f_i = e_{\beta(i)}^{F_{\alpha(i)}}$. There are $2^{O(\lambda \log(kb))}$ options. We perform a simple sanity check we consider only options where the mapping $i \mapsto (\alpha(i), \beta(i))$ is injective.

We also guess a partition of $\mathcal{B}$ into sets $\mathcal{B}_1, \ldots, \mathcal{B}_\kappa$, aiming at $F_j \in \mathcal{B}_j$. By standard objects for derandimizing color-coding, this can be turned into a branch into $2^{O(\kappa)} \log |\mathcal{B}|$ options.

For every $1 \le j \le \kappa$ and every $B \in \mathcal{B}_j$ we make a sanity check: we expect that, for every $i \in [\lambda]$ such that $\alpha(i) = j$ the edge $e_{\beta(i)}^B$ actually lies on P_i. If this is not the case, we remove B from $\mathcal{B}_j$ and $\mathcal{B}$ (making its edges undeletable). Clearly, F_j remains in $\mathcal{B}_j$ in the branch where the guesses are correct.

We now make the following (main) filtering step. Iterate over all $1 \le j \le \kappa$ and indices $1 \le i_1, i_2 \le \lambda$ such that $\alpha(i_1) = \alpha(i_2) = j$ but $\beta(i_1) < \beta(i_2)$. Consider $B \in \mathcal{B}_j$ such that there exists another $B' \in \mathcal{B}_j$ such that $e_{\beta(i_1)}^{B'}$ is before $a_{\beta(i_1)}^B$ on P_{i_1} but $e_{\beta(i_2)}^{B'}$ is after $a_{\beta(i_2)}^B$ on P_{i_2}. Assume $B = F_j$. Then $B' \cap Z = \emptyset$. By the property of the Weighted Bundled Cut with Order instance, G contains a path Q from an endpoint of $e_{\beta(i_1)}^{B'}$ to an endpoint of $e_{\beta(i_2)}^{B'}$ that uses only undeletable edges and edges of B'. Hence, Q is disjoint with Z. However, as $e_{\beta(i_1)}^B \in Z$ lies after $e_{\beta(i_1)}^{B'}$ on P_{i_1} while $e_{\beta(i_2)}^B \in Z$ lies before $e_{\beta(i_2)}^{B'}$ on P_{i_2}, Q goes from the s-side of Z to the t-side of Z, a contradiction. Hence, $B \neq F_j$ and we can remove such B from $\mathcal{B}_j$ and $\mathcal{B}$.

After we perform the filtering step exhaustively, for every $1 \le j \le \kappa$ we can order $\mathcal{B}_j$ as $B_{j,1}, \ldots, B_{j,|\mathcal{B}_j|}$ such that for every $1 \le \xi < \zeta \le |\mathcal{B}_j|$ and for every $i \in [\lambda]$, if $\alpha(i) = j$, then $e_{\beta(i)}^{B_\xi}$ is before $e_{\beta(i)}^{B_\zeta}$ on P_i. Let a_j^Z be such that $F_j = B_{j,a_j^Z}$.

In the end, we also check if still $\lambda = \lambda_G(s,t)$. If this is not the case, we terminate the current branch.

An edge $e \in E(G)$ is *vulnerable* if $e = e_{\beta(i)}^B$ for some $i \in [\lambda]$ and $B \in \mathcal{B}_{\alpha(i)}$. Note that all edges of Z are vulnerable, all vulnerable edges are deletable, and vulnerable edges appear only on paths of $\mathcal{P}$.

We now construct an auxiliary weighted directed graph H as follows. Start with H consisting of two vertices s and t. For every $1 \le j \le \kappa$, add a path P_j^H from s to t with $|\mathcal{B}_j|$ edges; denote the a-th edge as $e_{j,a}$ and set its weight as $\omega(e_{j,a}) = \omega(B_{j,a})$. Furthermore, for every $1 \le i_1, i_2 \le \lambda$, denote $j_1 = \alpha(i_1)$, $j_2 = \alpha(i_2)$, for every $1 \le a_1 \le |\mathcal{B}_{j_1}|$ and $1 \le a_2 \le |\mathcal{B}_{j_2}|$, for every endpoint u_1 of $e_{\beta(i_1)}^{B_{j_1,a_1}}$, for every endpoint u_2 of $e_{\beta(i_2)}^{B_{j_2,a_2}}$, if G contains a path from u_1 to u_2 consisting only of nonvulnerable edges, then add to H an edge of weight $+\infty$ from the corresponding endpoint of e_{j_1,a_1} (i.e., tail if and only if u_1 is a tail of $e_{\beta(i_1)}^{B_{j_1,a_1}}$) to the corresponding endpoint of e_{j_2,a_2} (i.e., tail if and only if u_2 is a tail of $e_{\beta(i_2)}^{B_{j_2,a_2}}$).

Observe that $Z' := \{e_{j,a_j^Z} \mid 1 \le j \le \kappa\}$ is an st-cut in H. Indeed, if H would contain an arc (v,u) with v before e_{j,a_j^Z} and u after $e_{j',a_{j'}^Z}$ for some $j,j' \in [\kappa]$, then this arc was added to H because of some path between the corresponding endpoints in G and such a path would lead from the s-side to t-side of Z. Also, in the other direction, observe that if $Y' = \{e_{j,a_j} \mid 1 \le j \le \kappa\}$ is an st-mincut in H, then

$$Y = \left\{ e_{\beta(i)}^{B_{\alpha(i),a_{\alpha(i)}}} \mid i \in [\lambda] \right\}$$

is a solution to $\mathcal{I}$ of the same weight.

Hence, it suffices to find in H an st-cut of cardinality κ and minimum possible weight. Since $\kappa \le \lambda_H(s,t)$, this can be done in polynomial time. ◻

3.2 Applications

In all three discussed problems, we are given both a bound k on the cardinality of the solution and a bound W on the total weight of the solution. The parameter is k.

In Weighted Cut, the solution is an st-cut. In Weighted DFAS, the solution is a feedback arc set in a directed graph. Note that by standard reduction, this is equivalent to Weighted DFVS. In Weighted b-Chain SAT, the problem is in fact a Weighted Bundled Cut where every bundle is a path of length at most b.

Note that Weighted Cut is Weighted Bundled Cut with bundles of size 1 and no undeletable edges while Weighted b-Chain SAT is Weighted Bundled Cut with bundles of size at most b. Furthermore, in both problems the assumption for Weighted Bundled Cut with Order is satisfied (in the latter case, order the edges along the path). We infer the following.

Theorem 3.3. *Weighted Cut is randomized FPT when parameterized by k.*

Theorem 3.4. *Weighted b-Chain SAT is randomized FPT when parameterized by k and b.*

By standard approach, Weighted DFAS can be solved using a subroutine for Weighted Skew Multicut. Here, we are given a directed graph G, a tuple $(s_i,t_i)_{i=1}^b$ of terminal pairs, a weight function $\omega : E(G) \to \mathbb{Z}_+$, and integers k, W. The goal is to find a set $Z \subseteq E(G)$ of cardinality at most k, weight at most W, and such that there is no path from s_i to t_j in $G - Z$ for any $1 \le i \le j \le b$. We observe the following reduction.

Lemma 3.5. *Given a Weighted Skew Multicut instance $\mathcal{I} = (G,(s_i,t_i)_{i=1}^b,\omega,k,W)$, one can in polynomial time construct an equivalent Weighted Bundled Cut with Order instance $\mathcal{I}' = (G',\mathcal{B},\omega,k,W)$ with the same k and W and bundles of size b each.*

Proof. To construct the graph G', we start with b disjoint copies $G^1,\dots,G^b$ of the graph G. By v^i, e^i, etc., we denote the copy of vertex v or edge e in the copy G^i. We set $B_e = \{e^i \mid i \in [b]\}$ for $e \in E(G)$ and $\mathcal{B} = \{B_e \mid e \in E(G)\}$, that is, all b copies of one edge of G form a bundle. We set weights of bundles as $\omega'(B_e) = \omega(e)$. There will be no more bundles, so all arcs introduced later to G' are undeletable.

For every $1 \le i < j \le b$ and $v \in V(G)$, we add to G' an arc (v^i,v^j). These arcs make the instance satisfy the "order" property with the natural order $(e^1,e^2,\dots,e^b)$ for the bundle B_e. Furthermore, we add to G' vertices s and t and arcs (s,s_i^i) and (t_i^i,t) for every $i \in [b]$. This finishes the description of the instance $\mathcal{I}' = (G',\mathcal{B},\omega',k,W)$.

It is straightforward to observe that if Z is a solution to the instance $\mathcal{I}$, then $\bigcup_{e\in Z} B_e$ is a solution to $\mathcal{I}'$ of the same weight and touching $|Z|$ bundles. In the other direction, note that if Z' is a solution to $\mathcal{I}'$ then $Z = \{e \in E(G) \mid Z' \cap B_e \neq \emptyset\}$ is a solution to $\mathcal{I}$. ◻

We deduce the following.

Theorem 3.6. *Weighted DFAS and Weighted DFVS are randomized FPT when parameterized by k.*

4 WEIGHTED ALMOST 2-SAT

The final application of the flow-augmentation framework is to the problem Weighted Almost 2-SAT. Let us recall the problem definition. The input consists of a 2-CNF formula ϕ, viewed as a set of 2-clauses; a weight function $\omega \colon \phi \to \mathbb{Z}_+$; and integers $k, W \in \mathbb{Z}_+$. The objective is to find a clause deletion set $Z \subseteq \phi$ such that $\phi - Z$ is satisfiable, $|Z| \le k$, and $\omega(Z) = \sum_{C\in Z}\omega(C) \le W$. We refer to a clause set $Z \subseteq \phi$ such that $\phi - Z$ is satisfiable as a *solution*. Our goal is thus to find such a solution Z, with $|Z| \le k$ and $\omega(Z) \le W$, in time FPT parameterized by k. We show the following.

Theorem 4.1. *Weighted Almost 2-SAT is randomized FPT parameterized by k, with a running time of $2^{k^{O(1)}} n^{O(1)}$.*

As in the Almost 2-SAT-algorithm of Kratsch and Wahlström [12], we solve the problem by reduction to the auxiliary problem Digraph Pair Cut. Let use define the weighted version of this. An instance $\mathcal{I} = (G,s,t,\omega,\mathcal{T},k,W)$ of Weighted Digraph Pair Cut consists of a digraph G with special vertices s and t, a weight function $\omega \colon E(G) \to \mathbb{Z}_+$, a set $\mathcal{T} \subseteq \binom{V(G)}{2}$ of pairs of vertices of G, and integers $k, W \in \mathbb{Z}_+$. Refer to a set of edges $Z \subseteq E(G)$ as a *solution* if Z is an st-cut and for every pair $p \in \mathcal{T}$, at most one endpoint of p is reachable from s in $G - Z$. The goal is then to find a solution $Z \subseteq E(G)$ such that $|Z| \le k$ and $\omega(Z) \le W$. Note that the addition of a sink vertex t differs from the original formulation of Digraph Pair Cut [12]; we add such a vertex t in order to make use of the flow-augmentation technique. We show that Weighted Almost 2-SAT FPT-reduces to Weighted Digraph Pair Cut.

Lemma 4.2. *Let $\mathcal{I} = (\phi,\omega,k,W)$ be an instance of Weighted Almost 2-SAT. In time $2^{O(k)}$ we can produce a list of instances $\mathcal{I}' = (G,s,t,\omega',\mathcal{T},k',W')$ of Weighted Digraph Pair Cut such that $\mathcal{I}$ is a yes-instance if and only if at least one instance $\mathcal{I}'$ is a yes-instance, where $k' = O(k)$ for each produced instance.*

Therefore, it suffices to provide an FPT-algorithm for Weighted Digraph Pair Cut. Let $\mathcal{I} = (G,s,t,\omega,\mathcal{T},k,W)$ be an instance of Weighted Digraph Pair Cut and let Z be a minimal solution to $\mathcal{I}$. Then Z must be a star st-cut; i.e., for every $(u,v) \in Z$, u is reachable from s in $G - Z$ but v is not. Therefore, the tool of flow augmentation for star st-cuts (Theorem 2.6) is applicable to the problem, and our algorithm is based around this method. We defer to the full version [9] for the proof.

REFERENCES

[1] Jianer Chen, Yang Liu, Songjian Lu, Barry O'Sullivan, and Igor Razgon. 2008. A fixed-parameter algorithm for the directed feedback vertex set problem. *J. ACM* 55, 5 (2008), 21:1–21:19. https://doi.org/10.1145/1411509.1411511

[2] Rajesh Chitnis, Marek Cygan, MohammadTaghi Hajiaghayi, Marcin Pilipczuk, and Michal Pilipczuk. 2016. Designing FPT Algorithms for Cut Problems Using Randomized Contractions. *SIAM J. Comput.* 45, 4 (2016), 1171–1229. https://doi.org/10.1137/15M1032077

[3] Rajesh Chitnis, László Egri, and Dániel Marx. 2017. List H-Coloring a Graph by Removing Few Vertices. *Algorithmica* 78, 1 (2017), 110–146. https://doi.org/10.1007/s00453-016-0139-6

[4] Rajesh Hemant Chitnis, Marek Cygan, Mohammad Taghi Hajiaghayi, and Dániel Marx. 2015. Directed Subset Feedback Vertex Set Is Fixed-Parameter Tractable. *ACM Trans. Algorithms* 11, 4 (2015), 28:1–28:28. https://doi.org/10.1145/2700209

[5] Rajesh Hemant Chitnis, László Egri, and Dániel Marx. 2013. List H-Coloring a Graph by Removing Few Vertices. In *Algorithms - ESA 2013 - 21st Annual European Symposium, Sophia Antipolis, France, September 2-4, 2013. Proceedings (Lecture Notes in Computer Science, Vol. 8125)*, Hans L. Bodlaender and Giuseppe F. Italiano (Eds.). Springer, 313–324. https://doi.org/10.1007/978-3-642-40450-4_27

[6] Rajesh Hemant Chitnis, MohammadTaghi Hajiaghayi, and Dániel Marx. 2013. Fixed-Parameter Tractability of Directed Multiway Cut Parameterized by the Size of the Cutset. *SIAM J. Comput.* 42, 4 (2013), 1674–1696. https://doi.org/10.1137/12086217X

[7] Marek Cygan, Pawel Komosa, Daniel Lokshtanov, Marcin Pilipczuk, Michal Pilipczuk, Saket Saurabh, and Magnus Wahlström. 2021. Randomized Contractions Meet Lean Decompositions. *ACM Trans. Algorithms* 17, 1 (2021), 6:1–6:30. https://doi.org/10.1145/3426738

[8] Marek Cygan, Daniel Lokshtanov, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. 2019. Minimum Bisection Is Fixed-Parameter Tractable. *SIAM J. Comput.* 48, 2 (2019), 417–450.

[9] Eun Jung Kim, Stefan Kratsch, Marcin Pilipczuk, and Magnus Wahlström. 2021. Directed flow-augmentation. *CoRR* abs/2111.03450 (2021). arXiv:2111.03450 https://arxiv.org/abs/2111.03450

[10] Eun Jung Kim, Stefan Kratsch, Marcin Pilipczuk, and Magnus Wahlström. 2021. Solving hard cut problems via flow-augmentation. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, Dániel Marx (Ed.). SIAM, 149–168. https://doi.org/10.1137/1.9781611976465.11

[11] Stefan Kratsch, Shaohua Li, Dániel Marx, Marcin Pilipczuk, and Magnus Wahlström. 2020. Multi-budgeted Directed Cuts. *Algorithmica* 82, 8 (2020), 2135–2155. https://doi.org/10.1007/s00453-019-00609-1

[12] Stefan Kratsch and Magnus Wahlström. 2020. Representative Sets and Irrelevant Vertices: New Tools for Kernelization. *J. ACM* 67, 3 (2020), 16:1–16:50. https://doi.org/10.1145/3390887

[13] Daniel Lokshtanov, M. S. Ramanujan, and Saket Saurabh. 2018. When Recursion is Better than Iteration: A Linear-Time Algorithm for Acyclicity with Few Error Vertices. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, Artur Czumaj (Ed.). SIAM, 1916–1933. https://doi.org/10.1137/1.9781611975031.125

[14] Daniel Lokshtanov, M. S. Ramanujan, Saket Saurabh, and Meirav Zehavi. 2020. Parameterized Complexity and Approximability of Directed Odd Cycle Transversal. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, Shuchi Chawla (Ed.). SIAM, 2181–2200. https://doi.org/10.1137/1.9781611975994.134

[15] Dániel Marx. 2004. Parameterized Graph Separation Problems. In *Parameterized and Exact Computation, First International Workshop, IWPEC 2004, Bergen, Norway, September 14-17, 2004, Proceedings (Lecture Notes in Computer Science, Vol. 3162)*, Rodney G. Downey, Michael R. Fellows, and Frank K. H. A. Dehne (Eds.). Springer, 71–82. https://doi.org/10.1007/978-3-540-28639-4_7

[16] Dániel Marx. 2006. Parameterized graph separation problems. *Theor. Comput. Sci.* 351, 3 (2006), 394–406. https://doi.org/10.1016/j.tcs.2005.10.007

[17] Dániel Marx, Barry O'Sullivan, and Igor Razgon. 2013. Finding small separators in linear time via treewidth reduction. *ACM Trans. Algorithms* 9, 4 (2013), 30:1–30:35. https://doi.org/10.1145/2500119

[18] Dániel Marx and Igor Razgon. 2009. Constant ratio fixed-parameter approximation of the edge multicut problem. *Inf. Process. Lett.* 109, 20 (2009), 1161–1166. https://doi.org/10.1016/j.ipl.2009.07.016

[19] Dániel Marx and Igor Razgon. 2014. Fixed-Parameter Tractability of Multicut Parameterized by the Size of the Cutset. *SIAM J. Comput.* 43, 2 (2014), 355–388. https://doi.org/10.1137/110855247

[20] Marcin Pilipczuk and Magnus Wahlström. 2018. Directed Multicut is $W[1]$-hard, Even for Four Terminal Pairs. *ACM Trans. Comput. Theory* 10, 3 (2018), 13:1–13:18. https://doi.org/10.1145/3201775

[21] Igor Razgon and Barry O'Sullivan. 2009. Almost 2-SAT is fixed-parameter tractable. *J. Comput. Syst. Sci.* 75, 8 (2009), 435–450. https://doi.org/10.1016/j.jcss.2009.04.002

[22] Saket Saurabh. 2017. What's next? Future directions in parameterized complexity. https://rapctelaviv.weebly.com/uploads/1/0/5/3/105379375/future.pdf Recent Advances in Parameterized Complexity school, Tel Aviv, December 2017.