Solving hard cut problems via flow-augmentation *

Eun Jung Kim † Stefan Kratsch ‡ Marcin Pilipczuk § Magnus Wahlström ¶

Abstract

We present a new technique for designing fixed-parameter algorithms for graph cut problems in undirected graphs, which we call *flow augmentation*. Our technique is applicable to problems that can be phrased as a search for an (edge) (s,t)-cut of cardinality at most k in an undirected graph G with designated terminals s and t.

More precisely, we consider problems where an (unknown) solution is a set $Z \subseteq E(G)$ of size at most k such that

- in $G - Z$, s and t are in distinct connected components,
- every edge of Z connects two distinct connected components of $G - Z$, and
- if we define the set $Z_{s,t} \subseteq Z$ as those edges $e \in Z$ for which there exists an (s,t)-path P_e with $E(P_e) \cap Z = \{e\}$, then $Z_{s,t}$ separates s from t.

We prove that in the above scenario one can in randomized time $k^{\mathcal{O}(1)}(|V(G)| + |E(G)|)$ add a number of edges to the graph so that with probably at least $2^{-\mathcal{O}(k \log k)}$ no added edge connects two components of $G - Z$, and $Z_{s,t}$ becomes a *minimum cut* between s and t.

This additional property becomes a handy lever in applications. For example, consider the question of an (s,t)-cut of cardinality at most k and of minimum possible weight (assuming edge weights in G). While the problem is NP-hard in general, it easily reduces to the maximum flow / minimum cut problem if we additionally assume that k is the minimum possible cardinality of an (s,t)-cut in G. Hence, we immediately obtain that the aforementioned problem admits an $2^{\mathcal{O}(k \log k)} n^{\mathcal{O}(1)}$-time randomized fixed-parameter algorithm.

We apply our method to obtain a randomized fixed-parameter algorithm for a notorious "hard nut" graph cut problem we call *Coupled Min-Cut*. This problem emerges out of the study of FPT algorithms for Min CSP problems (see below), and was unamenable to other techniques for parameterized algorithms in graph cut problems, such as Randomized Contractions, Treewidth Reduction or Shadow Removal.

In fact, we go one step further. To demonstrate the power of the approach, we consider more generally the Boolean Min CSP(Γ)-problems, a.k.a. Min SAT(Γ), parameterized by the solution cost. This is a framework of optimization problems that includes problems such as Almost 2-SAT and the notorious ℓ-Chain SAT problem. We are able to show that every problem Min SAT(Γ) is either (1) FPT, (2) W[1]-hard, or (3) able to express the soft constraint $(u \to v)$, and thereby also the min-cut problem in directed graphs. All the W[1]-hard cases were known or immediate, and the main new result is an FPT algorithm for a generalization of Coupled Min-Cut. In other words, flow-augmentation is powerful enough to let us solve every fixed-parameter tractable problem in the class, except those that explicitly encompass directed graph cuts.

1 Introduction

Fixed-parameter tractable algorithms for graph separation problems has been an important question in parameterized complexity, and after more than a decade of intense study it would seem that we should by now know of all the major techniques necessary for the design of such algorithms. Certainly, there is an impressive toolbox, leading to the resolution of central problems such as FPT algorithms for Multicut [26, 2] and Minimum Bisection [9].

Yet despite this progress, several open problems remain. Many of these relate to directed graph cuts, such as the existence of FPT algorithms for the notorious ℓ-Chain SAT problem identified by Chitnis et al. [4], and the deceptively simple-looking problem of Bi-objective (s,t)-cut [20]. In the former, the input is a digraph $D = (V, A)$ with distinguished vertices s, t and a budget k, where the arcs of A are partitioned into *chains* $\{(v_1 \to v_2), (v_2 \to v_3), \ldots, (v_{\ell-1} \to v_\ell)\}$ on at most $\ell = O(1)$ vertices, and the task is to find an (s,t)-cut that consists of arcs of at most k chains. In particular, ℓ-Chain SAT has been identified as a problem of central importance, since Egri et al. [4] showed that its resolution is the central missing piece for a dichotomy of fixed-parameter tractability of a natural parameterization of the List H-Coloring class of problems. Bi-objective (s,t)-cut is even simpler to describe. The input is a digraph $D = (V, A)$ with arc weights w and $s, t \in V$, and two budgets k, W, and the task is to find an (s,t)-cut $Z \subseteq A$ such that $|Z| \leq k$ and $w(Z) \leq W$. Again, despite the simplicity of the problem, the existence of an FPT algorithm is open.

Another open problem comes from the study of parameterized aspects of *constraint satisfaction problems* (CSPs; see below), although the problem can be readily phrased as a graph problem. We dub this graph problem

*Eun Jung Kim is supported by the grant from French National Research Agency under JCJC program (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 (M. Pilipczuk).

†Université Paris-Dauphine, PSL Research University, CNRS, UMR 7243, LAMSADE, 75016, Paris, France.

‡Humboldt-Universität zu Berlin, Germany.

§University of Warsaw, Warsaw, Poland.

¶Royal Holloway, University of London, TW20 0EX, UK.

Coupled Min-Cut. The input is a graph $G = (V, E)$ with vertices $s, t \in V$ and a budget k, where the edges of G are (sub)partitioned into *pairs*, and the task is to find an (s,t)-cut $Z \subseteq E$ consisting of at most k pairs, where furthermore for every edge pair (e_1, e_2) not in Z, at most one of the two edges is reachable from s in $G - Z$. Although this is a problem about unweighted, undirected graph cuts, it has been completely resistant to attacks by the existing toolbox of graph separation problems.[1]

To understand the difficulty, it is helpful to consider two variants of the problem. First, if you remove the coupling between the edges (i.e., if the task is simply to find an (s,t)-cut consisting of at most k pairs), the result is a well-known W[1]-hard problem, first identified by Marx and Razgon [25]. On the other hand, if the coupling is strengthened to require that precisely one edge is reachable from s in $G - Z$, or if the pairs are replaced by connected sets $\{v_1v_2, v_2v_3, v_3v_4\}$ of edges, then the result is readily solved by existing methods (perhaps the easiest existing method is to use the treewidth reduction of Marx et al. [24]). Coupled Min-Cut strikes a balance between these variants which makes it very difficult to handle.

We introduce the technique of *flow augmentation* for the construction of FPT algorithms for graph separation problems. To illustrate and justify it, consider one of the three problems above and assume that we knew that the solution Z had to be an (s,t)-*min*-cut, i.e., of minimum cardinality. Then an FPT-algorithm for each of the three above-mentioned problems reduces to a nice exercise. We omit details for now.

The idea of the flow augmentation technique is to take a given graph $G = (V, E)$ with vertices $s, t \in V$ and an unknown (s,t)-cut Z (that corresponds to the solution to your problem) and add edges A to G in a way such that with probability at least $1/f(k)$, the new edges A do not connect two distinct connected components of $G - Z$ and Z is an (s,t)-min cut in the resulting graph $G + A$. As it happens, we are only able to show flow augmentation for undirected graphs, and the resolution of Coupled Min-Cut takes quite a bit more work beyond a single flow augmentation application, but this discussion hopefully illustrates why a flow augmentation procedure is a useful goal.

Beyond the tractability of Coupled Min-Cut, we show the following:

1. A randomized procedure for flow augmentation, discussed next;
2. the definition of a problem, Generalized Coupled MinCut (GCMC), that generalizes Coupled Min-Cut into a "maximal tractable problem" (in some sense), and an FPT algorithm for GCMC that makes heavy use of flow augmentation;
3. a study of Min SAT(Γ) (see Section 1.2), showing that each such problem is either (1) FPT, (2) W[1]-hard, or (3) captures directed graph cuts, and is hence out of scope for our present work.

The essential new tractable case of Min SAT(Γ) is represented by GCMC, whereas all W[1]-hard cases are easy or previously known. Hence, at least for CSP-style optimization problems, flow augmentation represents the last missing technique in the toolbox for undirected graph cut problems.

1.1 The flow augmentation technique The central result of our paper is the following tool. Consider an undirected graph $G = (V, E)$ with two vertices $s, t \in V$, and an unknown (s,t)-cut Z. Furthermore, let $Z_{s,t} \subseteq Z$ be those edges with one endpoint reachable from s and the other reachable from t in $G - Z$. We say that Z is a *special (s,t)-cut* if $Z_{s,t}$ is an (s,t)-cut, and *eligible for* (s,t) if additionally every edge of Z has its endpoints in different connected components of $G - Z$. In particular, any minimal, not necessarily minimum (s,t)-cut is eligible for (s,t). Another example of an eligible (s,t)-cut, important in the study of Boolean Min CSP problems, is a star cut. A *star (s,t)-cut* is a set of edges Z such that Z is an (s,t)-cut and every edge of Z has precisely one endpoint reachable from s in $G - Z$. Again, clearly a star (s,t)-cut is eligible for (s,t).

Let $k = |Z|$, $\lambda^* = |Z_{s,t}|$, and let $\lambda_G(s,t) \leq \lambda^*$ be the value of an (s,t)-max flow in G. We show the following:

Theorem 1.1. *There is a randomized algorithm that, given an undirected graph $G = (V, E)$ with $s, t \in V$ and two integers $k \geq \lambda^* \geq \lambda_G(s,t)$, in time $k^{\mathcal{O}(1)}(|V| + |E|)$ outputs an edge multiset A with $\lambda_{G+A}(s,t) \geq \lambda^*$ and a flow $\mathcal{P}$ in $G + A$ of cardinality λ^*, such that for any (s,t)-cut Z in G eligible for (s,t) with $|Z| = k$ and $|Z_{s,t}| = \lambda^*$, with probability $2^{-\mathcal{O}(k \log k)}$, the following holds: for every $uv \in A$, u and v are connected in $G - Z$; and for every path $P \in \mathcal{P}$, $|E(P) \cap Z| = 1$.*

In particular, in any successful run, in $G+A$ the paths $\mathcal{P}$ will be an (s,t) max-flow, $Z_{s,t}$ will be an (s,t)-min cut, and this information (plus the explicit paths $\mathcal{P}$) can be useful for cleaning up the problem.

While Theorem 1.1 and most of the statements in this

[1] Although the question of an FPT algorithm was never asked in a public forum, the problem was known to the community after quickly having been identified as an obstacle to the study of parameterized algorithms for Min CSP.

paper claim *randomized* algorithms, they are all easy to derandomize: all randomized steps are either color-coding steps (derandomizable by standard tools, see e.g. [7]) or in fact plain branching steps. For sake of clarity of the arguments, we present them as randomized algorithms and refrain from discussing derandomization.

Example applications. To illustrate the flow-augmentation technique, consider the following two example problems. Both these problems admit FPT algorithms through other methods (e.g., Randomized Contractions [3]), although the flow-augmentation-based algorithms are particularly simple to give.

First, recall the Bi-objective (s,t)-Cut problem. Papadimitriou and Yannakakis showed that this is strongly NP-hard, even for undirected graphs, and also showed partial approximation hardness [27]. The directed version, with $\ell \geq 2$ distinct budgets, was recently considered from a parameterized perspective by Kratsch et al. [20], who showed that the problem is FPT if all budgets are included in the parameter, but W[1]-hard if at least two budgets k_i are not included in the parameter. The case of a single budget not being included in the parameter, which includes the Bi-objective (s,t)-Cut problem parameterized by k, is open. Although its FPT status was left open, Kratsch et al. [20] were able to show that this problem is well-behaved in the sense that the number of distinct extremal ("closest") solutions in a certain sense is bounded by $g(k)$, for some function g.

If k equals the minimum cardinality of an (s,t)-cut, the problem can be easily solved via any polynomial-time minimum cut algorithm: set the capacity of every edge to be a large number (much larger than any weight of an edge) plus the weight of an edge and ask for a minimum capacity cut. Hence, in *undirected graphs* a simple randomized FPT algorithm can be obtained as follows: We prepend the step above with flow-augmentation (Theorem 1.1), with newly added edges assigned prohibitively large weights, and repeat the process $2^{O(k \log k)}$ times.

Similarly, let us consider Edge Bipartization with both a cardinality budget k and a weight budget W, dubbed Bi-objective Edge Bipartization. In this problem, the input is an edge-weighted graph $G = (V, E)$ and two integers k, W, and the question is whether there is a set of edges $F \subseteq E$ such that $G - F$ is bipartite, $|F| \leq k$, and the total weight of F is at most W.

Flow-augmentation again gives a simple FPT algorithm for this problem (when parameterized by k). Let $F_0 \subseteq E$ be an edge bipartization set with $|F_0| = k$ but with no regard for weight; such a set can be computed in $2^k n^{\mathcal{O}(1)}$ time by an FPT algorithm for the unweighted problem [12]. It is at the core of the original iterative compression algorithm for Odd Cycle Transversal [32] that having access to a bipartization set F_0 allows us to convert the bipartization problem on G into the solution of $2^{\mathcal{O}(|F_0|)}$ cut problems; in our case, the same reduction lets us solve Bi-objective Edge Bipartization via the solution of $2^{\mathcal{O}(k)}$ instances of Bi-objective (s,t)-Cut.

A more complicated example is Edge Multicut. Here, the input consists of an undirected multigraph G, an integer k, and a family $T \subseteq \binom{V(G)}{2}$ of cut requests. The goal is to find a set X of at most k edges so that for every $st \in T$, s and t are in different connected components of $G - X$.

Fixed-parameter tractability of Edge Multicut, parameterized by k only (i.e., with unbounded number of cut requests) was a long-standing open question in parameterized complexity until 2010, when two groups of researchers [2, 26] announced a positive solution. The first solution [2] involves a deep study of the combinatorics of the problem with a highly problem-specific reduction rules simplifying the instance. The second solution [26], was significantly simpler thanks to a new technique called *Shadow Removal*, that turned out to be applicable to many other graph separation problems (e.g. [21, 5]).

Interestingly, Edge Multicut seems not to be amenable to a number of general frameworks for undirected graph separation problems, including Randomized Contractions [3] and Treewidth Reduction [24]. Up to now, Shadow Removal was the only general technique applicable to Edge Multicut.

We show that Edge Multicut can be also solved using flow-augmentation instead of shadow removal. The reduction[2] follows first the lines of the algorithm of Marx and Razgon [26] that reduces it, using only basic tools, to a variant dubbed Bipedal Multicut Compression. The second part reduces this variant to Coupled Min-Cut, thus showing applicability of flow-augmentation.

Flow-augmentation versus previous methods. To illustrate the need for the flow-augmentation framework, let us briefly review previous work on parameterized algorithms for graph separation problems. In this, we will review how these works fail to apply to the Coupled Min-Cut and more generally Generalized Coupled MinCut problems. We consider previous work in three categories.

Greedy methods and shadow removal. One of the more powerful methods of the area is the *shadow removal* method of Marx and Razgon [26], which is a way to

[2]The reduction is presented only as a motivation for the new technique and included for completeness. We do not claim the authorship of this reduction. While we are not aware of any citable source of this reduction, it has been floating around in the community in the last years.

randomly "clean up" a graph so that the solution is more well-behaved. This has been an important component of many results for directed and undirected graphs, e.g., [5].

Shadow removal builds on an earlier concept of *important separators*, due to Marx [23]. Both shadow removal and important separators build on a principle that some component of the solution could be chosen in a *greedy* manner; for example, that some cut in the graph, cutting away a component C from some set of terminals T, can be chosen to cut as close to the terminals as possible. This is frequently useful, and indeed was central to the FPT algorithm for Multicut by Marx and Razgon [26]; but the edge-coupling constraint in Coupled Min-Cut appears to prevent any such greedy strategy. For the same reason, these methods fail to apply for weighted problems such as Bi-objective (s,t)-Cut.

Graph decompositions. Another very successful strategy is to represent or "understand" the connectivity structure of a graph via some form of graph decomposition. This has been done in several ways. One of the earlier is the *treewidth reduction* method of Marx et al. [24], which gives a bounded treewidth decomposition of a projection of the graph that preserves all minimal (s,t)-cuts of size at most k.

Another variant on the decomposition theme is *recursive understanding*. Here, the input graph $G = (V, E)$ is decomposed along a sparse cut, say $Z = \delta(S)$ for $S \subseteq V$ where both $|S|$ and $|V \setminus S|$ are substantial, and the behavior of $G[S]$ with respect to the edges of Z is recursively "understood" so that $G[S]$ can be represented or simplified. This strategy was employed by Kawarabayashi and Thorup [16] for the k-Way Cut problem, and was simplified and sharpened by Chitnis et al. [3] via *randomized contractions*. See also the FPT algorithm for Minimum Bisection [9] which constructs a tree decomposition capturing *all* small cuts in a graph G [9, 8].

However, for Coupled Min-Cut, the coupling constraints prevent decomposition methods from being used. On the one hand, if a sparse cut is found in G, then the edge coupling implies that the two parts of the cut are not truly independent, and recursive methods do not seem to apply. On the other hand, if the coupling constraints were to be represented explicitly as another type of edges in the graph, then a solution to Coupled Min-Cut of size k would correspond to an (s,t)-cut of unbounded capacity.

We also remark that, in contrast to Theorem 1.1, Randomized Contractions introduce polynomial in the graph size factor in the running time bound that is far from being linear.

Relaxation-based methods. The category of methods which comes closest to flow augmentation is arguably the work on building FPT algorithms for optimization problems using well-behaved problem relaxations. The most famous is branching over half-integral LP-relaxations, which has been used in many of the most efficient FPT algorithms for optimization problems [10, 22, 14], but also other relaxations than LP-relaxations have been used to the same effect, e.g., flow-based relaxations for linear-time FPT algorithms [15, 13, 30].

These methods are related to flow augmentation in the general concept of solving a problem by FPT-reducing it to a tractable optimization problem. But these methods only work when there is a suitable tractable relaxation, which we are not aware of for Coupled Min-Cut. Relaxation-based methods also have not yet been applied to weighted problems, such as Bi-objective (s,t)-Cut above.

1.2 Parameterized complexity of Boolean Min CSP Let Γ be a finite set of Boolean relations, i.e., a finite Boolean *constraint language*. A *constraint* $R(X)$ over Γ is a pair of a relation $R \in \Gamma$ and a tuple X of variables, and it is *satisfied* by an assignment $\phi\colon X \to \{0,1\}$ if the tuple $\phi(X)$ is in R. A *formula over* Γ is a conjunction $\mathcal{F}$ of constraints over Γ. The problem Min SAT(Γ) for a Boolean constraint language Γ takes as input a formula $\mathcal{F}$ over Γ and an integer k, and asks whether there is an assignment ϕ such that all but at most k constraints in $\mathcal{F}$ are satisfied by ϕ. Note that both ℓ-Chain SAT and Coupled Min-Cut are examples of Min SAT(Γ) for specific languages Γ. The classical complexity of Min SAT(Γ) was characterized by Khanna et al. [17], and this result has recently been vastly generalized [19, 33]. We study the parameterized complexity of Min SAT(Γ) parameterized by k, for languages Γ which do not express directed graph cuts, i.e., languages which cannot express clauses $(u \to v)$. This is a natural restriction for us, since our result for flow augmentation only applies to undirected graphs. We show the following.

Theorem 1.2. *For every finite Boolean language Γ that does not express soft clauses $(u \to v)$, Min SAT(Γ) is either FPT or W[1]-hard (both by parameter k and $k + |\Gamma|$).*

The characterization uses mostly standard methods, with one new ingredient we refer to as *constraint coloring*. For a full description of the method and the complexity characterization behind the theorem see the full version of the paper [18]. We provide a brief sketch.

A *relational co-clone* is a set of relations closed under conjunction and existential quantification (so-called *pp-definitions*). Noting that we may assume that Γ cannot

express $(u \to v)$ even using pp-definitions, we as a starting point consider the maximal Boolean relational co-clones that exclude $(u \to v)$, of which there are four, as captured in Post's lattice [29, 6]. Under duality and a previously known hardness result [1], it suffices to consider two cases, which must be inspected more carefully. These correspond to (1) every $R \in \Gamma$ can be defined over the language $\{(x = 0), (x = 1), (x = y), (x \neq y)\}$, and (2) every $R \in \Gamma$ can be defined over the language $\{(x = 0), (x = 1), (x = y), (\neg x_1 \vee \ldots \vee \neg x_d)\}$ for some $d \in \mathbb{Z}$ (without using existential quantification).

In this sketch, we consider the simpler language $\Gamma_0 = \{(x = 0), (x = 1), (x = y)\}$. That is, consider a finite constraint language Γ such that every relation $R \in \Gamma$ can be defined as the set of solutions to a formula using constraints from Γ_0. Furthermore, from previous work [25] we know that a relation $R \in \Gamma$ such that $R(a, b, c, d) \equiv (a = b) \wedge (c = d)$ yields a W[1]-hard problem. We refer to R as *double equality*. Also define a tractable language $\Gamma_1 = \{(x = 0), (x = 1), (x = y), R_{0,1,=}\}$ where $R_{0,1,=}(a, b, c, d) \equiv (a = 0) \wedge (b = 1) \wedge (c = d)$. We use constraint coloring to show that for every language Γ over Γ_0 which does not express double equality, Min SAT(Γ) FPT-reduces to Min SAT(Γ_1).

Let $(\mathcal{F}, k)$ be an instance of Min SAT(Γ), and assume there is an assignment ϕ such that at most k constraints of $\mathcal{F}$ are false in ϕ. For every constraint $R(X)$ in $\mathcal{F}$, guess a random assignment α_R to X. Since $|\Gamma|$ is finite, with some probability $2^{-\mathcal{O}(k)}$, α_R agrees with ϕ for every constraint $R(X)$ that is false in ϕ. Assume this holds. Now assume that for some $x, y \in X$, a clause $(x = y)$ holds in both $R(X)$ and in α_R. We may then assume that $(x = y)$ holds in the optimal solution ϕ, and may identify x with y in $\mathcal{F}$, simplifying R. By similar steps, we can reduce any $R \in \Gamma$ that does not implement double equality to a constraint $R_{0,1,=}(X')$. Completing such an analysis over the languages mentioned above, we find that every problem Min SAT(Γ) that does not implement a variant of double equality reduces to (1) one of two relatively simple problems that can be solved by branching, and (2) the problem Generalized Coupled MinCut, described next.

1.3 Main new tractable case: Generalized Coupled MinCut

Our main algorithmic contribution is a new fixed-parameter tractable undirected graph separation problem Generalized Coupled MinCut (GCMC for short) that encapsulates the new isle of tractability in the aforementioned CSP dichotomy result.

The input to GCMC consists of:

- An undirected multigraph G with designated vertices $s, t \in V(G)$, $s \neq t$.
- A multiset $\mathcal{C}$ of pairs of vertices of $V(G) \setminus \{s, t\}$, called henceforth *pairs*.
- A family $\mathfrak{B}$ of disjoint subsets of $\mathcal{C} \uplus E(G)$, called henceforth *blocks*.
- An integer k.

An edge or a pair e is *soft* if it is contained in a block of $\mathfrak{B}$, and *crisp* otherwise. For a block $B \in \mathfrak{B}$, by $V(B) = \bigcup_{e \in B} e$ we denote the vertices involved in edges or pairs of B.

Fix an instance $\mathcal{I} = (G, \mathcal{C}, \mathfrak{B}, k)$ and consider a set $S \subseteq V(G)$. We say that an edge $e \in E(G)$ is *violated* by S if $e \in \delta(S)$, i.e. e has precisely one endpoint in S, and *satisfied* otherwise. Similarly, we say that a pair $p \in \mathcal{C}$ is *violated* by S if $p \subseteq S$, and *satisfied* otherwise. The notions of being violated and satisfied extend to blocks: a block is violated if it contains a violated edge or a violated pair, and satisfied otherwise.

A set $S \subseteq V(G)$ is a *solution* to the instance $\mathcal{I}$ if

- $s \in S$ but $t \notin S$, and
- no crisp edge or pair is violated by S.

The *cost* of a solution S is the number of violated blocks. The GCMC problem asks for a solution of cost at most k.

Let $\mathcal{I} = (G, \mathcal{C}, \mathfrak{B}, k)$ be a GCMC instance. We say that $\mathcal{I}$ is ***b*-bounded** for an integer **b** if every block of the instance is of size at most **b** (i.e., the number of edges and pairs in a single block is at most **b**). For a block $B \in \mathfrak{B}$, let G_B be the graph with vertex set $V(B) \setminus \{s, t\}$ such that two vertices u, v are adjacent in G_B if and only if $uv \in B$ (i.e., uv is an edge or a pair of B). Then, we say that $\mathcal{I}$ is $2K_2$*-free* if for every block B of the instance, G_B does not contain $2K_2$ (a four-vertex graph consisting of two edges with distinct endpoints) as an induced subgraph.

We are now ready to state the main algorithmic result.

Theorem 1.3. *The Generalized Coupled MinCut problem, restricted to $2K_2$-free **b**-bounded instances, is fixed-parameter tractable when parameterized by **b** and k.*

We give a brief sketch of our algorithm for GCMC, which is the main application of Theorem 1.1 in this paper. It can be readily assumed that the input graph G is connected, and the solution S we are chasing is connected in G and has solution cost precisely k. The number of violated clauses (both edge and pair) will be at most $\kappa := k\mathbf{b}$. The algorithm consists of a series of technical reductions and instance simplifications so that (with probability $1/f(\kappa)$) the sought solution S can be assumed to be molded to satisfy some structural description. To highlight the insight that guides these steps, we illustrate the case of Coupled Min-Cut introduced at the beginning of the section.

Let us begin with the toy case where the sought

solution Z has to be a min-(s,t)-cut whose cardinality equals λ. For a max-(s,t)-flow $\mathcal{P}$, each path P of $\mathcal{P}$ must intersect with Z precisely once and all vertices on P *before* (*after*) the edge of $P \cap Z$ are reachable from s (from t) in $G - Z$. We view each path $P \in \mathcal{P}$ orientation from s to t, and call these paths *flow-paths*. Observe that if a set of edges on flow-paths forms a directed cycle, where the orientation of each edge is decided by the direction of its flow-path, these edges must be contained in the same connected component of $G - Z$. Consequently, we can simplify the instance by contracting (a) all edges of G that are not on flow paths, and (b) all directed cycles of flow paths. The resulting flow-paths $\mathcal{P}$ are called *tidy*. Assume for simplicity that Z is also known to consist of k pairs of edges, i.e. $\lambda = 2k$. One can guess the 'coupling' of flow-paths, thereby dictating how the pairs of edges in Z should be located over the flow-paths; only the edges and pairs of edges of G which conform to this coupling will survive in the sense that all other edges will be contracted (or forbidden as *crisp* edge) and pairs will be unpaired. Now consider two paths $P, P' \in \mathcal{P}$ and two edge pairs $(e,e'),(f,f') \in E(P) \times E(P')$. The key fact here is that if (e,e') *dominates* (f,f') in that both e and e' come before f and f' on the respective paths P, P', then no solution Z will take the pair (f,f') as this will leave both edges of the pair (e,e') reachable from s in $G - Z$, which is forbidden by the problem definition. Therefore, we can make the edges f, f' crisp. Consequently, the set of all edge pairs between P and P' forms an *antichain*. Once we reach this streamlined picture, it can be easily verified that if $(uv, u'v')$ is an edge pair between P and P' such that u (resp. u') is before v (resp. v') on P (resp. P'), then u and v' are on the opposite sides of $G - Z$, i.e. $u \in R_s(Z)$ if and only if $v' \in R_t(Z)$ and the same holds for v and u'. Observe that now the requirement that at most one of paired edges are reachable from s in $G - Z$ will be automatically satisfied. Moreover, any Z which meets this new condition chooses uv if and only if it chooses $u'v'$. That is, we have reduced to finding an assignment $\phi : V(G) \to \{0,1\}$ with $\phi(s) = 1$ and $\phi(t) = 0$ under the *precedence condition* that the assigned value cannot increase along a flow-path (imposed by $|P \cap Z| = 1$ for each $P \in \mathcal{P}$), and the newly derived condition from edge pairs. This can be expressed as Almost 2-SAT, i.e. the problem of finding an assignment satisfying all but at most k' clauses of a given 2-SAT formula, which is fixed-parameter tractable by Razgon and O'Sullivan [31].

The full generality towards an algorithm for Generalized Coupled MinCut creates much more complication. Nevertheless, two ideas from the above illustration remain crucial. First, we reduce to an instance $\mathcal{I} = (G, \mathcal{C}, \mathfrak{B}, k)$ equipped with a partition of the edge multi-set $E(G)$ such that there is a total order on the edges of the same 'type'. Secondly, the total orders naturally induce *domination relations* which allow us to simplify the edge pairs and blocks.

Central to realizing these ideas is the notion of a *flow-tree decomposition*. Observe that a connected set S containing s but not t gives rise to what we dubbed as a *star* (s,t)*-cut*, namely an (s,t)-cut Z such that every $e \in Z$ has precisely one endpoint which is reachable from s in $G - Z$. Conversely, for a star (s,t)-cut Z of G the vertex set $R_s(Z)$ reachable from s in $G - Z$ satisfies $\delta(R_s(Z)) = Z$. Therefore, we can equivalently seek for a star (s,t)-cut of cardinality at most κ which violates at most k blocks. Because a star (s,t)-cut is a type of cut for which the flow-augmentation can be applied (see Section 2), we may assume with success probability $2^{-\mathcal{O}(\kappa \log \kappa)}$ that an augmenting set has been already been added to the graph (as crisp edges) and flow-paths $\mathcal{P}$ satisfying the condition of Theorem 1.1, called *a witnessing flow for* Z, is given. Note that now the edge multi-set $Z_{s,t} \subseteq Z$ with one endpoint in $R_s(Z)$ and another in $R_t(Z)$ is a minimum (s,t)-cut and we have flow-paths $\mathcal{P}$ witnessing this.

Consider a path P in $G - E(\mathcal{P})$ with endpoints in $V(\mathcal{P})$. Since $Z_{s,t}$ is an (s,t)-cut, the endpoints of P are either both in $R_s(Z)$ or both in $R_t(Z)$, hence these endpoints can be identified into a single vertex in another tidying procedure. Hence for tidy $\mathcal{P}$, every connected component $\hat{H}$ of $G - V(\mathcal{P})$ is adjacent with exactly one vertex s_H of $V(\mathcal{P})$, which we call *an attachment vertex*. We remark that we cannot identify the entire path P because, unlike the above toy case, connected components of $G - V(\mathcal{P})$ are now relevant for the remaining $Z \setminus Z_{s,t}$. If S intersects with $V(\hat{H}) \cup \{s_H\}$ in a nontrivial way (that is, S contains at least one vertex but does not contain the entire set, in which case we say $\hat{H}$ is *active*) and thus produces at least one violated edge in $H := G[V(\hat{H}) \cup \{s_H\}]$, then $S \cap V(H)$ yields again a star (s_H, t_H)-cut of H for some $t_H \in V(\hat{H}) \setminus S$, termed a *local sink*. Therefore, we can apply the flow-augmentation recursively to H and the subsequent connected components that appear along the way. Notice that if a component H is decomposed with a (tidy) flow-path $\mathcal{P}_H$, there will be more components created with attachment vertices on $V(\mathcal{P}_H)$. The newly created components will be naturally placed as 'children' of H. This leads to a canonical tree-structured decomposition, called a *flow-tree decomposition*.

While the precise definition of flow-tree decomposition and an recursive algorithm for constructing one

can be found in [18], two issues arise immediately. When do we proceed or stop to decompose a component, and how do we know the local sink? Regarding the first question, we construct a flow-tree decomposition in such a way that we need to be correct in proceeding with the decomposition only when a component is active, and in such a case the budget for violated edges decreases. We proceed until the 'depth' of the flow-tree reaches κ and assuming that we have been correct in this liberal sense, any component after this point (*leaf* of the flow-tree) can be declared inactive. Concerning the local sink, edge pairs with an endpoint in a component are the only reason why S can be potentially active. Therefore, the candidates for a local sink will be endpoints of such pairs, and an involved guessing procedure returns a local sink with sufficient probability.

Once we obtain a flow-tree decomposition which is generously wrong on inactive nodes, but correct on all active nodes (with probability $2^{-\kappa^{\mathcal{O}(1)}}$), we color-code the nodes of the flow-tree so that the active nodes are colorful with good probability. Now we may assume that Z induces a connected subtree of size $\mathcal{O}(\kappa)$ in the flow-tree via active nodes, and furthermore the nodes of the same color naturally yields the *types* of flow-paths ("i-th flow-path in a node colored by β"). Furthermore, inductively from the top flow-paths, we can give a linear order on the edges of a flow-path of the same type, which then induces a linear order on the nodes (of the same color) in the order of their attachment vertices on the parent flow-paths. This provides a canonical total order of the edges on the flow-paths of the same type.

In a similar (as in the toy example above) but much more involved way, the flow-tree structure allows to reduce groups of blocks into (appropriately defined) antichains. In the toy example, edges on one path are linearly ordered in such a way that in the sought solution a prefix of the order is contained in S. By a number of involved color-coding steps, we obtain the same "linear order" property on edges on paths of the same type, leading to an antichain property similar to the one in the toy example. As a result, we obtain again a reduction of the input instance to Almost 2-SAT.

1.4 Organization This extended abstract contains a description of the flow-augmentation technique, that is, an almost full proof (with a few proofs omitted due to space constraints) of Theorem 1.1. For other results mentioned in this introduction, we refer to the full version [18].

2 The flow-augmentation technique

In this section we develop our flow-augmentation technique. We focus here on problems that can be cast as (or reduced to) finding a certain (s,t)-cut in a given undirected multi-graph G. The key observation is that many such NP-hard cut problems become tractable or at least fixed-parameter tractable when the allowable cut size k matches the maximum (s,t)-flow in G. The core idea of our technique is to attempt to augment the maximum (s,t)-flow by adding additional edges, a so-called flow-augmenting set, while not breaking any candidate solution to the problem.

As an example, it is NP-hard to find an (s,t)-cut in an edge-weighted graph that is of minimum total weight and of cardinality at most k, but easily reduces to min cut / max flow computation if one assumes that k is the minimum cardinality of an (s,t)-cut in the graph. While the cost of a solution in this case is simply its cardinality, generally, the cost of a set of edges (solution) may be more complicated in other applications, e.g., edges may be paired up arbitrarily and we may delete the edges of up to ℓ pairs to separate s and t; in this case the cost of a solution edge set shall be the number ℓ of pairs whose union contains the set. Meanwhile, the cardinality of the cut is at most $k = 2\ell$. The question that will then be asked is, of course, whether maximum (s,t)-flow of exactly k makes the problem tractable or at least fixed-parameter tractable with respect to k. Such an algorithm implies fixed-parameter tractability with respect to the solution cost as long as we can derive an upper bound on the cardinality k in terms of the solution cost.

2.1 Preliminaries

2.1.1 Basic notation We consider only (finite) undirected *multi-graphs without loops*. In particular, different edges connecting the same pair of vertices are considered to be identifiable and non-interchangeable.[3] Formally, a *multi-graph* could be captured as $G = (V, E, \pi)$ where V and E are finite sets and $\pi\colon E \to \binom{V}{2}$ assigns each edge in E an unordered pair of endpoints. To keep notation within reason, we will treat multi-graphs as pairs $G = (V, E)$ where V is a finite set and E is a multi-subset of $\binom{V}{2}$ but understanding that cuts X (to be defined in a

[3] This generality seems necessary to cover a largest set of applications. Multiple copies of the same edge in G might arise in the reduction of some problem to an appropriate cut problem. The different copies may have wildly different behavior regarding contribution to solution cost. Our goal will be to ensure that all solutions of a certain cardinality in terms of cut size have a good probability of being preserved, thereby remaining oblivious to many unnecessary details of the application.

moment) could involve deleting particular (identifiable) copies of virtually the same edge $\{u, v\}$. For a multi-graph G and A a multi-set of edges on V, the graphs $G + A$ and $G - A$ are accordingly understood as starting from G and, respectively, adding all edges in A that are not yet in G or removing from G all edges that are also in A; again, note that this may include different edges with the same two endpoints. For a vertex set S, we denote by $\delta(S)$ the multi-set of edges that have precisely one endpoint in S, and by $\partial(S)$ the set of vertices in S that are incident with at least one edge in $\delta(S)$. By a *connected component* we mean a maximal set $S \subseteq V$ that induces a connected subgraph of G. In all other aspects we follow standard graph notation as set out by Diestel [11].

Throughout this paragraph let $G = (V, E)$ be an arbitrary multi-graph, let $S, T \subseteq V$, and let $X \subseteq E$. Define $R_S(X)$ as the set of vertices that are reachable from any vertex in S in $G - X$. The set X is an (S, T)-*cut* if $R_S(X) \cap R_T(X) = \emptyset$; note that no such cut exists if $S \cap T \neq \emptyset$. A *minimum* (S, T)-*cut* is any (S, T)-cut of minimum possible cardinality; whereas X is a *minimal* (S, T)-*cut* if no proper subset of X is an (S, T)-cut. (We will crucially need both minimum and minimal cuts.) By the well-known duality of cuts and flows in graphs (Menger's theorem suffices here) we get that the cardinality of any minimum (S, T)-cut is equal to the maximum number of edge-disjoint paths from S to T in G or, equivalently, to the maximum unit-capacity (S, T)-flow. By $\lambda_G(S, T)$ we denote the maximum flow from S to T or, equivalently, the minimum size of an (S, T)-cut in G; we omit the subscript G when it is clear from context. We mostly apply these notions for the special cases of $S = \{s\}$ and $T = \{t\}$ and then write, e.g., (s, t)-cut rather than $(\{s\}, \{t\})$-cut for succinctness. In particular, we write $\lambda_G(s, t)$ rather than $\lambda_G(\{s\}, \{t\})$ and, when G, s, and t are understood, we usually abbreviate this to λ. We say that an (S, T)-cut X is *closest to* S if for every other (S, T)-cut X' with $R_S(X') \subseteq R_S(X)$ we have $|X'| > |X|$. (This specializes the notion of closeness used in previous work to cuts.) Clearly, if X is an (S, T)-cut closest to S then X must in particular be minimal.

Let us recall two useful facts about edge cuts in graphs.

PROPOSITION 2.1. *Let X be a minimal (S, T)-cut. Then $X = \delta(R_S(X)) = \delta(R_T(X))$.*

PROPOSITION 2.2. *There is a unique minimum (S, T)-cut that is closest to S.*

2.1.2 Special cuts, eligible cuts, compatibility, and flow-augmentation. Let $G = (V, E)$ be a connected, undirected multi-graph, and let vertices $s, t \in V$. For $Z \subseteq E$, let $Z_{s,t} \subseteq Z$ be the set of edges with one endpoint in $R_s(Z)$ and one endpoint in $R_t(Z)$.

The following notions are crucial for this section.

DEFINITION 2.1. (SPECIAL CUT) *We say that an (s, t)-cut Z is* special *if $Z_{s,t}$ is an (s, t)-cut. That is, the set of edges $Z_{s,t} \subseteq Z$ with one endpoint in $R_s(Z)$ and one endpoint in $R_t(Z)$ is also an (s, t)-cut.*

Note that special (s, t)-cuts generalize minimal (s, t)-cuts.

In this section, we focus on solutions that are special (s, t)-cuts with an additional technical property.

DEFINITION 2.2. (ELIGIBLE CUT) *We say that an (s, t)-cut Z is* eligible for (s, t) *if*

1. *Z is special, and*
2. *each edge of Z has its endpoints in different connected components of $G - Z$.*

For an integer λ^*, we say that an (s, t)-cut Z is λ^*-eligible if Z is eligible and additionally $|Z_{s,t}| = \lambda^*$.

The next two definitions formalize two properties we want from a set of edges that we add to the graph: (i) it does not break the solution, and (ii) it increases the flow from s to t.

DEFINITION 2.3. (COMPATIBLE SET) *A multi-subset A of $\binom{V}{2}$ is* compatible *with a set $Z \subseteq E$ if for every $uv \in A$, u and v are connected in $G - Z$.*

DEFINITION 2.4. (FLOW-AUGMENTING SET) *For an integer $\lambda^* \geq \lambda_G(s, t)$, a multi-subset A of $\binom{V}{2}$ is* λ^*-flow-augmenting *if $\lambda_{G+A}(s, t) \geq \lambda^*$.*

Intuitively, the role of Z will be played by an unknown solution to the cut problem in question and compatibility of A with Z means that A cannot add connectivity that was removed by Z (or that was not present in the first place). The challenge is to find a flow-augmenting set that with good probability is consistent with at least one solution Z, without knowing Z beforehand.

It will be convenient to take edges in A as being *undeletable* or, equivalently, as unbounded (or infinite) capacity. Clearly, if A is flow-augmenting and compatible with an (eligible) set Z then A remains flow-augmenting and compatible with Z after adding an arbitrary number of copies of any edges in A. In particular, having a total of $k + 1$ copies of every edge in A will make those edges effectively undeletable for sets Z of size k, that is, the endpoints of any edge in A cannot be separated by Z. Note that for applications, since edges in A are in addition to the original input, one will usually not be interested in deleting edges of A anyway (and costs may not be

defined), and they only help to increase the flow to match an (unknown) solution. For the purpose of flow and path packings, edges in A may, accordingly, be shared by any number of (flow) paths, fully equivalent to simply having $k+1$ copies of each edge.

2.1.3 Witnessing flow. To simplify for applications, in addition to returning a flow-augmenting set, we will also attempt to return an (s,t)-max flow in the augmented graph which intersects $Z_{s,t}$ in a particularly structured way.

In the following, let G be a connected graph with $s,t \in V(G)$, and let Z be an (s,t)-cut in G which contains an (s,t)-min cut. A *witnessing (s,t)-flow for Z* in G is an (s,t)-max flow $\mathcal{P}$ in G such that every edge of $Z_{s,t}$ occurs on a path of $\mathcal{P}$, and every path of $\mathcal{P}$ intersects Z in precisely one edge.

We make a few observations. First, since Z is an (s,t)-cut, every (s,t)-path in G intersects Z in at least one edge. Second, if additionally $\lambda_G(s,t) = |Z_{s,t}|$, then every (s,t)-max flow in G is witnessing *for* $Z_{s,t}$. Hence, if Z is a minimum (s,t)-cut, then finding a witnessing flow is no harder than finding a flow-augmenting set. However, if Z is a special and only $Z_{s,t}$ is a minimum (s,t)-cut, then a witnessing flow is a more restrictive notion.

We now observe that for every special (s,t)-cut Z, one can augment G with a set compatible with Z such that $Z_{s,t}$ becomes a (s,t)-min cut and $G+A$ admits a witnessing flow for G.

LEMMA 2.1. *Let $G=(V,E)$ be a multi-graph, let $s,t \in V$ with $s \neq t$, let $Z \subseteq E$ be a special (s,t)-cut of size k, and let $\lambda^* = |Z_{s,t}|$. Then there exists a λ^*-flow-augmenting set A compatible with Z and a witnessing flow $\mathcal{P}$ for Z in $G+A$.*

Proof. For each pair u and v of vertices in the same connected component of $G-Z$, add to A a set of $k+1$ copies of the edge uv. Clearly, A is compatible with Z. For every $e=uv \in Z_{s,t}$ with $u \in R_s(Z)$ and $v \in R_t(Z)$, let P_e be a path in $G+A$ consisting of the edges $su \in A$, $uv \in Z_{s,t}$, and $vt \in A$. Then, $\mathcal{P} := \{P_e \mid e \in Z_{s,t}\}$ is a witnessing flow for Z in $G+A$ of cardinality λ^*. Hence, A is λ^*-flow-augmenting. ∎

A few remarks are in place. The proof of Lemma 2.1 shows that a set $Z \subseteq E$ admits a λ^*-flow-augmenting set A if and only if Z does not contain an (s,t)-cut of cardinality less than λ^*. Indeed, in one direction such a cut $C \subseteq Z$ remains an (s,t)-cut in $G+A$, preventing the flow from increasing above $|C|$, and in the other direction the set A constructed in the proof of Lemma 2.1 is in some sense "maximum possible" and all (s,t)-cuts of cardinality at most k in $G+A$ are contained in Z. Furthermore, even if Z is a special (s,t)-cut where $Z_{s,t}$ is an (s,t)-min cut (so no flow increase is possible), while Z may not admit a witnessing flow in G, it is possible to augment G with a set of edges compatible with Z so that a witnessing flow exists.

Lemma 2.1 motivates the following extension of the definition of compatibility.

DEFINITION 2.5. (COMPATIBLE PAIR) *A pair $(A,\mathcal{P})$ is* compatible *with a special (s,t)-cut Z if A is a λ^*-flow-augmenting set compatible with Z for $\lambda^* = |Z_{s,t}|$ and $\mathcal{P}$ is a witnessing flow for Z in $G+A$.*

2.1.4 Problem formulation. The proof of Lemma 2.1 shows that the task of finding a compatible flow-augmenting set and a witnessing flow would be trivial if only we knew Z in advance. Not knowing Z, we will have to place additional edges more sparingly than in the proof of Lemma 2.1 to arrive at a sufficient success probability. Let us formally define our goal, taking into account that the set Z is not known.

In the FLOW-AUGMENTATION SAMPLING problem we are given an instance (G,s,t,k,λ^*) consisting of an undirected multi-graph $G=(V,E)$, vertices $s,t \in V$, and integers k and λ^* such that $k \geq \lambda^* \geq \lambda := \lambda_G(s,t)$. The goal is to find (in probabilistic polynomial-time) a multi-set A of $\binom{V}{2}$ and an (s,t)-flow $\mathcal{P}$ in $G+A$ such that the following holds:

- $\lambda_{G+A}(s,t) \geq \lambda^*$, $|\mathcal{P}| = \lambda^*$, and
- for each λ^*-eligible (s,t)-cut Z of size exactly k, the output $(A,\mathcal{P})$ is compatible with Z with probability at least p.

The function p (that may depend on k or λ) is called the *success probability*.

In order to relax some corner cases, we allow for the event that $\lambda_{G+A}(s,t) > \lambda^*$, and note that if Z is an eligible (s,t)-cut with $|Z_{s,t}| = \lambda^*$ then for any such output $(A,\mathcal{P})$ such that A is compatible with Z we must have $\lambda_{G+A}(s,t) = \lambda^*$.

We begin the proof of Theorem 1.1 by introducing an appropriate decomposition of (the vertex set of) G into what we call *bundles*, which in turn consist of what is called *blocks*. We then present our recursive flow-augmentation algorithm, splitting the presentation into an "outer loop" and an "inner loop." Note that we assume that the input multi-graph G is connected as this somewhat simplifies presentation, but we will circumvent this assumption in applications.

It will be convenient to assume that we only care about λ^*-eligible cuts that do not contain any edge

incident with s nor t. This can be easily achieved by adding an extra terminal s' connected with s with $k+1$ edges, adding an extra terminal t' connected with t with $k+1$ edges, and asking for (s',t')-cuts instead. Consequently, in the proof we can assume one more property of an λ^*-eligible (s,t)-cut Z:

3. Z contains no edge incident with s or t.

2.2 Blocks and bundles Given an instance (G,s,t,k,λ^*) of FLOW-AUGMENTATION SAMPLING, it should come as no surprise that the minimum (s,t)-cuts of G will be crucial for flow-augmentation. Recall, however, that even structurally simple graphs may exhibit an exponential number of possibly crossing minimum (s,t)-cuts. We will use the notion of closest cuts (and implicitly the well-known uncrossing of minimum (s,t)-cuts as used in Proposition 2.2) to identify a sequence of non-crossing minimum (s,t)-cuts. The parts between consecutive cuts will be called blocks; we will also define a partition of blocks into consecutive groups called bundles. The decomposition of G into bundles will guide the choice of edges for the flow-augmenting set A in our algorithm and will be used to capture parts of G to recurse on.

For convenience, let us fix an instance (G,s,t,k,λ^*) and let $\lambda := \lambda_G(s,t) \leq k$ for use in this subsection. Accordingly, in G there is a packing of λ edge-disjoint (s,t)-paths $P_1,\ldots,P_\lambda$ (and no larger packing exists). Clearly, every minimum (s,t)-cut in G contains exactly one edge from each path P_j and no further edges. As noted earlier, we assume for now that G is connected.

2.2.1 Blocks We first define a sequence $C_0,\ldots,C_p$ of non-crossing minimum (s,t)-cuts; recall that minimum (s,t)-cuts in G all have cardinality λ. To start, let C_0 be the unique minimum (s,t)-cut that is closest to s. Inductively, for $i \geq 1$, let C_i be the minimum (s,t)-cut closest to s among all cuts that fulfil $N[R_s(C_{i-1})] \subseteq R_s(C_i)$. The cut C_i is well-defined (i.e., unique) by an easy variant of Proposition 2.2: Minimum cuts X fulfilling the requirement that $N[R_s(C_{i-1})] \subseteq R_s(X)$ uncross into minimum cuts fulfilling the same requirement. Intuitively, the construction is equivalent to asking that each C_i is closest to s among minimum (s,t)-cuts that do not intersect $C_0 \cup \ldots \cup C_{i-1}$ but this would need a formal proof and we do not require it.

We can now define the *blocks* $V_0,\ldots,V_{p+1} \subseteq V$, which will be seen to form a partition of V. Block V_0 is simply set to $R_s(C_0)$. For $i \in \{1,\ldots,p\}$, we define block V_i as the set of vertices reachable from s in $G - C_i$ but not in $G - C_{i-1}$, i.e., $V_i := R_s(C_i) \setminus R_s(C_{i-1})$. Finally, V_{p+1} contains all vertices reachable from s in G but not in $G - C_p$ which, since G is connected, equates to $V_{p+1} = V \setminus R_s(C_p)$. By construction of the cuts C_i we clearly have $s \in R_s(C_0) \subsetneq R_s(C_1) \subsetneq \ldots \subsetneq R_s(C_p) \subseteq V \setminus \{t\}$, so the blocks V_i are all nonempty and clearly form a partition of V.

Let us point out that blocks V_i do not need to be connected even though G is connected. It will be useful to note, however, that blocks V_0 and V_{p+1} are connected: The graph G is connected and each minimum (s,t)-cut C_i will therefore separate it into exactly two connected components $R_s(C_i)$ and $R_t(C_i)$. Blocks $V_0 = R_S(C_0)$ and $V_{p+1} = V \setminus R_s(C_p) = R_t(C_p)$ are therefore connected. Moreover, each block is at least somewhat connected through subpaths of the flow paths $P_1,\ldots,P_\lambda$ that are contained therein. We establish a bit more structure via the following two propositions.

PROPOSITION 2.3. *For each (s,t)-flow path $P_j \in \{P_1,\ldots,P_\lambda\}$, seen as being directed from s to t, the edges of the minimum (s,t)-cuts $C_0,\ldots,C_p$ appear in order of the cuts. These edges define a partition of the flow path P_j into $P_j^0,\ldots,P_j^{p+1}$ so that P_j^i is contained in block V_i for $i \in \{0,\ldots,p+1\}$.*

Using the fact that, for each (s,t)-flow path P_j, the blocks V_i contain consecutive subpaths of P_j, we can prove that each block has at most λ connected components. Moreover, each such component in a block V_i, with $i \in \{1,\ldots,p\}$ is incident with some number of edges of C_{i-1} and the same number of edges in C_i.

PROPOSITION 2.4. *Each block V_i has at most λ connected components. Moreover, each connected component in a block V_i, with $i \in \{1,\ldots,p\}$, is incident with $c \geq 1$ edges in C_{i-1} and with exactly c edges in C_i. (Clearly, V_0 is incident with all λ edges of C_0, and V_{p+1} is incident with all λ edges of C_p.)*

It can be easily verified that the decomposition into blocks can be computed in polynomial time.

PROPOSITION 2.5. *Given a multi-graph $G = (V,E)$ and vertices $s,t \in V$, the unique sequence of cuts $C_0,\ldots,C_p$ and decomposition of blocks $V_0,\ldots,V_{p+1}$ can be computed in polynomial time.*

2.2.2 Bundles We will now inductively define a decomposition of V into *bundles* $W_0,\ldots,W_{q+1}$. The first bundle W_0 is simply equal to the (connected) block V_0, which contains s. For $i \geq 1$, supposing that blocks $V_0,\ldots,V_{j-1}$ are already parts of previous bundles,

- let $W_i := V_j$ if V_j is connected (i.e., if $G[V_j]$ is connected) and call it a *connected bundle*
- otherwise, let $W_i := V_j \cup \ldots \cup V_{j'}$ be the union of contiguous blocks, where j' is maximal such that $G[V_j \cup \ldots \cup V_{j'}]$ is *not* connected and call it a *disconnected bundle.*

Observe that the final bundle is $W_{q+1} = V_{p+1}$ because V_{p+1} is connected and, due to the included subpaths of (s,t)-flow paths (cf. Proposition 2.4), any union $V_j \cup \ldots \cup V_{p+1}$ induces a connected graph (see also Proposition 2.6). We use $\mathsf{block}(W_i)$ to denote the set of blocks whose union is equal to W_i, i.e., $\mathsf{block}(W_i) = \{V_j\}$ and $\mathsf{block}(W_i) = \{V_j, \ldots, V_{j'}\}$ respectively in the two cases above. We say that two bundles W_i and $W_{i'}$ are *consecutive* if $|i - i'| = 1$.

Intuitively, bundles are defined as maximal sequences of blocks that permit a good argument to apply recursion in our algorithm. In case of a single block, if we augment the edges incident with the block, then in the recursive step the cardinality of the maximum flow $\lambda_G(s,t)$ increases. In case of a union of contiguous blocks that does not induce a connected subgraph, if we recurse into every connected component independently, we split the budget k in a nontrivial way, as every connected component contains the appropriate part of at least one flow path of $\mathcal{P}$.

Clearly, the bundles $W_0, \ldots, W_{q+1}$ are well defined and they form a partition of the vertex set V of G. We emphasize that $W_0 = V_0 \ni s$ and $W_{q+1} = V_{p+1} \ni t$ and that they are both connected bundles. We note without proof that the bundles inherit the connectivity properties of blocks because the cuts between blocks combined into a bundle connect their subpaths of (s,t)-flow paths $P_1, \ldots, P_\lambda$ into longer subpaths, whereas the incidence to the preceding and succeeding cuts stays the same (see Proposition 2.6). For ease of reference, let us denote by $C'_0, \ldots, C'_q$ those cuts among $C_0, \ldots, C_p$ that have endpoints in two different (hence consecutive) bundles, concretely, with C'_i having endpoints in both W_i and W_{i+1}; note that $C'_0 = C_0$ as $W_0 = V_0$ and $C'_q = C_p$ as $W_{q+1} = V_{p+1}$.

PROPOSITION 2.6. *Each bundle W_i has at most λ connected components. Moreover, each connected component in a bundle W_i, with $i \in \{1, \ldots, q\}$, is incident with $c \geq 1$ edges in C'_{i-1} and with c edges in C'_i. (Clearly, $W_0 = V_0$ is incident with all λ edges of $C'_0 = C_0$, and $W_{q+1} = V_{p+1}$ is incident with all λ edges of $C'_q = C_p$.)*

Let us introduce some more notation for bundles: For $0 \leq a \leq b \leq q+1$ let $W_{a,b} := \bigcup_{i=a}^{b} W_i$. Let $W_{\leq a} := W_{0,a}$ and $W_{\geq a} := W_{a,q+1}$. For any (union of consecutive bundles) $W_{a,b}$ we define the *left interface* $\mathsf{left}(W_{a,b})$ as $\partial(W_{\geq a}) \cap W_{\geq a}$ when $a \geq 1$ and as $\{s\}$ when $a = 0$. (I.e., when $a \geq 1$ then $\mathsf{left}(W_{a,b})$ are those vertices of $W_{a,b}$ that are incident with the cut C'_{a-1} that precedes bundle W_a). Similarly, we define the *right interface* $\mathsf{right}(W_{a,b})$ as $\partial(W_{\leq b}) \cap W_{\leq b}$ when $b \leq q$ and as $\{t\}$ when $b = q+1$. (I.e., when $b \leq q$ then $\mathsf{right}(W_{a,b})$ are those vertices of $W_{a,b}$ that are incident with the cut C'_b that succeeds bundle W_b.) For single bundles W_i the same notation applies using $W_i = W_{i,i}$. A consecutive subsequence of bundles is called a *stretch* of bundles, or simply a stretch.

While a union of consecutive blocks may be disconnected, this is not true for bundles where, as can be easily checked, any two consecutive bundles together induce a connected subgraph of G.

PROPOSITION 2.7. *For any two consecutive bundles W_i and W_{i+1} the graph $G[W_i \cup W_{i+1}]$ is connected.*

Clearly, the decomposition into bundles can be efficiently computed from the one into blocks.

PROPOSITION 2.8. *Given a multi-graph $G = (V, E)$ and vertices $s, t \in V$, the unique sequence of cuts $C'_0, \ldots, C'_q$ and decomposition of bundles $W_0, \ldots, W_{q+1}$ can be computed in polynomial time.*

2.2.3 Affected and unaffected bundles We will later need to reason about the interaction of a special (s,t)-cut $Z \subseteq E$ and $G = (V, E)$ and, hence, about the interaction with the bundles of G. We say that a bundle W is *unaffected by Z* if $N[W]$ is contained in a single connected component of $G - Z$; otherwise we say that W is *affected by Z*. As an example, the cut $Z = C'_i$ affects both W_i and W_{i+1} but no other bundles. Similarly, a cut Z entirely confined to $G[W_i]$ affects only W_i, since $W_{\leq i-1}$ and $W_{\geq i+1}$ are both connected and disjoint from Z. The more interesting/difficult cuts Z affect several bundles in a non-trivial way.

The following observation limits the number and arrangement of affected bundles. It will be important for reducing the general case (probabilistically) to the case where G decomposes into a bounded number of bundles. Concretely, this is the purpose of the outer-loop part of our algorithm, which is presented in the following section.

LEMMA 2.2. *Let $Z \subseteq E$ be an (s,t)-cut of size at most k. Let $0 \leq a \leq b \leq q+1$ and let ℓ be the number of indices $a \leq i \leq b$ such that the bundle W_i is affected. Then,*

$$\ell \leq 2\,|W_{a-1,b+1} \cap Z|\,.$$

In particular, at most $2k$ bundles are affected by Z.

LEMMA 2.3. *Let $Z \subseteq E$ be an (s,t)-cut of size at most k. There is at most one maximal stretch $W_{a,b}$ of bundles such that every bundle W_i, $a \leq i \leq b$ is affected by Z and such that $W_{a,b}$ contains both a vertex reachable from s and a vertex reachable from t (in $G - Z$). Moreover, all vertices in the left interface of $W_{a,b}$ are reachable from s and all vertices in the right interface are reachable from t. Finally, if Z is a special (s,t)-cut then there must be such a stretch.*

The previous lemma says that each special (s,t)-cut Z yields exactly one maximal stretch of affected bundles in which it separates s from t (and possibly creates further connected components). We say that Z *strongly affects* that stretch. For all other maximal affected stretches of bundles we say that they are *weakly affected by* Z. Note that a non-special cut such as $C'_i \cup C'_j$ for $j \geq i+3$ may contain no strongly affected stretch.

Let us make some useful observations about bundles not in the strongly affected stretch.

PROPOSITION 2.9. *Let Z be a special (s,t)-cut and let $W_{a,b}$ be the unique strongly affected stretch. Then the following hold.*

1. *For every $i < a$, if W_i is an unaffected bundle then $W_i \subseteq R_s(Z)$*
2. *For every $i > b$, if W_i is an unaffected bundle then $W_i \subseteq R_t(Z)$*
3. *If $W_{i,j}$ is a (maximal) weakly affected stretch with $j < a$, then $\mathsf{left}(W_i) \cup \mathsf{right}(W_j) \subseteq R_s(Z)$ and $(j-i+1) \leq 2|Z \cap W_{i,j}|$*
4. *If $W_{i,j}$ is a (maximal) weakly affected stretch with $i > b$, then $\mathsf{left}(W_i) \cup \mathsf{right}(W_j) \subseteq R_t(Z)$ and $(j-i+1) \leq 2|Z \cap W_{i,j}|$*

2.3 The outer loop of the algorithm Our algorithm SAMPLE consists of an outer loop (to be explained in this section), which is applied first to an input instance (G, s, t, k, λ^*) and also to certain instances in recursive calls, and an inner loop, which is applied only to short sequences of bundles. The outer loop part uses a color-coding approach to guess weakly and strongly affected stretches of bundles in G, and calls the inner-loop subroutine called SHORT-SEPARATION on the latter. This subroutine (to be described in detail in the following section) then seeks to recursively find an output $(A, \mathcal{P})$, using the assumption that whenever it is called on a stretch $W_{a,b}$, then either Z is disjoint from the stretch $W_{a,b}$ or $W_{a,b}$ is precisely the unique strongly affected stretch in G.

Each call to our algorithm will return a pair $(A, \mathcal{P})$ for the instance in question, where $(A, \mathcal{P})$ may or may not be compatible for an arbitrary (unknown) (s,t)-cut Z. A crucial observation for the correctness of our algorithm is that any flow-augmentation set guessed for an unaffected stretch of bundles will always be compatible with Z. This allows us to focus our attention in the analysis on the guesses made while processing affected bundles. This is essential in bounding the success probability purely in terms of k.

We will argue that for some sufficiently large constants $c_1 \gg c_2 \gg 0$, SAMPLE(G, s, t, k, λ^*) returns an output $(A, \mathcal{P})$ which is, with probability at least $e^{-g(\lambda_G(s,t),k)}$, compatible with an (unknown) eligible (s,t)-cut Z, where $g(\lambda, k) = (c_1 k - c_2)(1 + \ln k) + c_2 \max(0, k - \lambda)$.

The main (outer loop) algorithm is shown in Figure 1.

2.3.1 Interface of the inner loop algorithm The inner-loop algorithm expects as input an instance $(G', s', t', k', \lambda')$ that has two additional properties and will return a pair $(A', \mathcal{P}')$. A *valid input* $(G', s', t', k', \lambda')$ *for the inner loop algorithm* has the following properties:

1. The graph G' decomposes into bundles $W'_0, \ldots, W'_{q+1}$, with $1 \leq q \leq 2k'$, and such that $W'_0 = \{s'\}$ and $W'_{q+1} = \{t'\}$. If $q = 1$, then we say that the instance is a *single-bundle* instance, otherwise if $q > 1$ it is a *multiple-bundle* instance.
2. We have $\lambda_{G'}(s', t') < \lambda' \leq k'$, i.e., the maximum (s', t')-flow in G' is lower than the target flow value λ' after augmentation.

Furthermore, let Z' be an (s', t')-cut in G'. We say that Z' is a *valid cut for* $(G', s', t', k', \lambda')$ if the following hold.

1. Z' is an eligible (s', t')-cut in G' with $|Z'| = k$ and $|Z'_{s,t}| = \lambda'$;
2. Z' affects precisely the bundles $W'_1, \ldots, W'_q$ in G'

In the following section we will describe a realization of this interface by two algorithms called SHORT-SEPARATION-SINGLE and SHORT-SEPARATION with the following success guarantee:

- for a valid single-bundle instance $(G', s', t', k', \lambda')$, the algorithm SHORT-SEPARATION-SINGLE returns a flow-augmenting set A' with $\lambda_{G'+A'}(s', t') \geq \lambda'$ and an (s,t)-flow $\mathcal{P}'$ in $G + A$ of size λ' such that for every valid cut Z', $(A', \mathcal{P}')$ is compatible with Z' with probability at least $32 \cdot e^{-g(\lambda_{G'}(s',t'),k')}$;
- for a valid multiple-bundle instance $(G', s', t', k', \lambda')$, the algorithm SHORT-SEPARATION returns a flow-augmenting set A' with $\lambda_{G'+A'}(s', t') \geq \lambda'$ and an (s,t)-flow $\mathcal{P}'$ in $G + A$ of size λ' such that for every valid cut Z', $(A', \mathcal{P}')$ is compatible with Z' with probability at least $32(k')^3 \cdot e^{-g(\lambda_{G'}(s',t'),k')}$.

2.3.2 Correctness of the outer loop part We are now ready to prove correctness of the outer loop algo-

Algorithm SAMPLE(G, s, t, k, λ^*)

1. If it does not hold that $\lambda_G(s,t) \leq \lambda^* \leq k$, then set A to be $\max(k+1, \lambda^*)$ copies of $\{s,t\}$, $\mathcal{P}$ to be any λ^* of these copies, and return $(A, \mathcal{P})$.
2. Initialize $A = \emptyset$ and $\mathcal{P}$ to be a set of λ^* zero-length paths starting in s.
3. Compute the partition $V = W_0 \cup \ldots \cup W_{q+1}$ of G into bundles.
4. Go into single mode or multiple mode with probability $1/2$ each.
   - In single mode, set $p_{\text{blue}} = p_{\text{red}} = 1/2$.
   - In multiple mode, set $p_{\text{blue}} = 1/k$, $p_{\text{red}} = 1 - 1/k$.
5. Randomly color each bundle blue or red; blue with probability p_{blue} and red with probability p_{red}.
6. Randomly sample an integer $\lambda^* \leq k' \leq k$ as follows: set $k' = k$ with probability $1/2$ and with remaining probability sample $\lambda^* \leq k' < k$ uniformly at random.
7. For every maximal stretch $W_{a,b}$ of bundles colored with the same color, do the following in consecutive order starting with $a = 0$, and maintaining the property that at the begining of the loop $\mathcal{P}$ is a family of λ^* edge-disjoint paths in $G + A$ starting in s and ending in $\mathsf{left}(W_a)$:
   (a) If $a > 0$, then add to A all edges uv for $u, v \in \mathsf{right}(W_{a-1}) \cup \mathsf{left}(W_a)$; (We henceforth refer to the edges added in this step as *link edges*.)
   (b) If the stretch is colored red and consists of one bundle in single mode, or at least two and at most $2k'$ bundles in multiple mode, then perform the following:
      i. Let G' be the graph $G[N[W_{a,b}]]$ with vertices of $W_{\leq a-1}$ contracted to a single vertex s' and vertices of $W_{\geq b+1}$ contracted to a single vertex t'. If $a = 0$, and hence $W_{\leq a-1} = \emptyset$, then instead add a new vertex s' and connect it to $s \in W_a$ via λ parallel edges $\{s, s'\}$. Similarly, if $b = q+1$ then $W_{\geq b+1} = \emptyset$ and we instead add a new vertex t' and connect it to $t \in W_b$ via λ parallel edges $\{t, t'\}$. Observe that $\deg(s') = \deg(t') = \lambda$.
      ii. Do a recursive call:
         - In single mode, let $(A', \mathcal{P}') \leftarrow$ SHORT-SEPARATION-SINGLE$(G', s', t', k', \lambda^*)$.
         - In multiple mode, let $(A', \mathcal{P}') \leftarrow$ SHORT-SEPARATION$(G', s', t', k', \lambda^*)$.
      iii. Update A as follows:
         - Add to A all edges of A' that are not incident with s' or t'.
         - For every edge $s'v \in A'$, add to A a separate edge uv for each vertex $u \in \mathsf{right}(W_{\leq a-1})$. If $a = 0$ then ignore edges $s's \in A'$ and for each edge $s'v \in A'$ add sv to A;
         - Analogously, for every edge $vt' \in A'$, add to A a separate edge vw for each vertex $w \in \mathsf{left}(W_{b+1})$. If $b = q+1$ then ignore edges $tt' \in A'$ and for each edge $vt' \in A'$ add vt to A.
      iv. Update $\mathcal{P}$ as follows: For every path $P' \in \mathcal{P}'$, if the first or last edge of P' belongs to A', replace it with one of its corresponding edges in A, and then pick a distinct path $P \in \mathcal{P}$ and append P' at the end of P, using a link edge to connect the endpoints of P and P' if necessary.
   (c) Otherwise:
      i. Add to A, with multiplicity $k+1$, all edges $\{u, w\}$ with $u \in \mathsf{right}(W_{a-1})$, taking $u = s$ if $a = 0$, and $w \in \mathsf{left}(W_{b+1})$, taking $w = t$ if $b = q+1$.
      ii. Prolong every path $P \in \mathcal{P}$ with a link edge (if $a > 0$) and an edge of A, so that P ends in $\mathsf{left}(W_{b+1})$, or in t if $b = q+1$.

Figure 1: The outer loop algorithm

rithm SAMPLE assuming a correct realization of the inner loop algorithm according to the interface stated above.

It is straightforward to verify the invariant stated in the loop: at every step, $\mathcal{P}$ is a family of λ^* edge-disjoint paths in $G + A$, starting in s and ending in $\mathsf{left}(W_a)$. It is also straightforward to verify the feasibility of the updates of $\mathcal{P}$. Furthermore, observe that after the last iteration of the loop, all paths of $\mathcal{P}$ end in t. Thus, at the end of the algorithm $\mathcal{P}$ is indeed a family of λ^* edge-disjoint paths from s to t in $G + A$.

We now prove that, in a well-defined sense, most edges in the returned set A are compatible with most minimal (s,t)-cuts Z.

LEMMA 2.4. *Let $W_{a,b}$ be a stretch processed by* SAMPLE *such that every bundle of the stretch is unaffected by Z. Then every edge added to A while processing $W_{a,b}$ is compatible with Z.*

Now we are set to prove correctness of the outer loop algorithm assuming a correct realization of the inner-loop interface.

LEMMA 2.5. *Assume that an algorithm* SHORT-SEPARATION *correctly realizes the above interface such that for every valid single-bundle (multiple-bundle) instance* $(G', s', t', k', \lambda')$ *with* $k' \leq k$*, the returned pair* $(A', \mathcal{P}')$ *is compatible with a fixed valid cut* Z' *with probability* $32e^{-g(\lambda_{G'}(s',t'),k')}$ $(32(k')^3 e^{-g(\lambda_{G'}(s',t'),k')})$*. Then for any* (G, s, t, k, λ^*)*,* SAMPLE *returns an* (s,t)*-flow-augmenting set* A *such that* $\lambda_{G+A}(s,t) \geq \lambda^*$ *and for any eligible* (s,t)*-cut* Z *in* G *of size* k *and with* $|Z_{s,t}| = \lambda^*$*, the returned pair* $(A, \mathcal{P})$ *is compatible with* Z *with probability at least* $e^{-g(\lambda_G(s,t),k)}$*.*

Proof. The lemma holds essentially vacuously if SAMPLE(G, s, t, k, λ^*) stops at step 1. Hence we assume $\lambda \leq \lambda^* \leq k$. Since G is connected, $\lambda \geq 1$, hence $k \geq 1$.

We first prove that all calls to SHORT-SEPARATION or SHORT-SEPARATION-SINGLE are made for valid instances $(G', s', t', k, \lambda^*)$. Let $(G', s', t', k, \lambda^*)$ be an instance on which SHORT-SEPARATION or SHORT-SEPARATION-SINGLE is called and let $W_{a,b}$ be the stretch that the call corresponds to. It can be verified that G', relative to minimum (s', t')-cuts, decomposes into bundles $\{s'\}, W_a, \ldots, W_b, \{t'\}$. A key point here is that s' and t' are both incident with precisely λ edges in G', and $\lambda_{G'}(s', t') = \lambda$. This makes $\delta(s')$ the unique closest minimum (s', t')-cut. From this point on, the sequence of closest minimum (s', t')-cuts that define blocks and bundles is identical to ones between the blocks that form bundles $W_a, \ldots, W_b$ in G. Clearly, $G'[W_a \cup \ldots W_b] \cong G[W_a \cup \ldots \cup W_b]$ (canonically) so we arrive at the same decomposition into bundles. At the end, $\delta(t')$ can be seen to be final closest minimum (s', t')-cut that arises when computing blocks and bundles for (G', s', t'), using a symmetric argument to the one for $\delta(s')$.

Now, we show the compatibility property. Let Z be any λ^*-eligible (s,t)-cut of size k. By Lemma 2.3, there is a unique strongly affected stretch $W_{a,b}$, and by Lemma 2.2 at most $2|Z|$ bundles are affected in total. Let $\ell = b - a + 1$ be the number of bundles in $W_{a,b}$ and let $Z' = Z \cap W_{a,b}$. We have $\ell \leq 2|Z'|$ and $\lambda^* \leq |Z'| \leq k$.

We are interested in the following success of the random choices made by the algorithm: the algorithm goes into mode single if $a = b$ and into mode multiple otherwise, $k' = |Z'|$, and the coloring of bundles in the loop is such that every bundle of $W_{a,b}$ is red, while W_{a-1}, W_{b+1}, and every other affected bundle is blue. Since there are at most $2(k - |Z \cap W_{a,b}|)$ affected bundles that are not in $W_{a,b}$, the above success happens with probability at least

- if $a = b$ and $k = |Z'|$: 2^{-5};
- if $a = b$ and $k > |Z'|$:

$$2^{-5}(k - \lambda^*)^{-1} 2^{-2(k-|Z'|)} \geq 2^{-5} k^{-1} 2^{-2(k-|Z'|)};$$

- if $a < b$:

$$\begin{aligned}(k - \lambda^* + 1)^{-1} \cdot k^{-2-2(k-|Z'|)} \cdot (1 - 1/k)^{\ell} \\ \geq k^{-3-2(k-|Z'|)} \cdot (1 - 1/k)^{2k} \\ \geq 2^{-4} k^{-3-2(k-|Z'|)}.\end{aligned}$$

Henceforth we assume that the above success indeed happens.

If this is the case, then for every two consecutive bundles W_i and W_{i+1} of different colors, either W_i or W_{i+1} is unaffected. In particular, all endpoints of the edges of $E(W_i, W_{i+1})$ are in the same connected component of $G - Z$. Thus, all link edges added to A are compatible with Z.

Let us now consider the processing of some maximal monochromatic stretch $W_{c,d}$ other than $W_{a,b}$. If $W_{c,d}$ is red, then by assumption on the coloring it is a stretch of unaffected bundles, and any edges added are compatible with Z by Lemma 2.4. Furthermore, any flow $\mathcal{P}'$ does not intersect Z, so the edges appended in the paths of $\mathcal{P}$ are disjoint with Z.

If $W_{c,d}$ is red, then we claim that $\mathsf{left}(W_c) \cup \mathsf{right}(W_d)$ are contained in the same connected component in $G - Z$. Indeed, by assumption on the coloring, any affected bundle in $W_{c,d}$ is contained in some weakly affected stretch $W_{c',d'}$ where the stretch is contained in $W_{c,d}$ in its entirety. By Prop. 2.9 the endpoints of such a stretch are contained in the same component of $G - Z$, as are the endpoints of any stretch of unaffected bundles. The claim follows. Thus the edges added by SAMPLE for $W_{c,d}$ are compatible with Z. Furthermore, in this case all edges appended to the paths of $\mathcal{P}$ are from A.

Now consider the strongly affected stretch $W_{a,b}$. Observe that SAMPLE will make a recursive call to SHORT-SEPARATION or SHORT-SEPARATION-SINGLE for this stretch; let the resulting instance be $(G', s', t', k', \lambda^*)$. Note that Z' are the edges of Z contained in G' and that Z' is a valid cut for $(G', s', t', k', \lambda^*)$. Furthermore, $\lambda = \lambda_G(s,t) = \lambda_{G'}(s', t')$. Indeed, by Lemma 2.3 $\mathsf{left}(W_a) \subseteq R_s(Z)$ and $\mathsf{right}(W_b) \subseteq R_t(Z)$, and since W_{a-1} (if any) and W_{b+1} (if any) are unaffected, these are entirely contained in $R_s(Z)$ respectively $R_t(Z)$ as well. Hence Z' is an eligible (s', t')-cut in G'. Finally, $|Z'| = k'$ and $|Z'_{s',t'}| = |Z_{s,t}| = \lambda^*$, and by assumption Z' affects every bundle W_i, $1 \leq i \leq q$, of G'.

Thus, since SHORT-SEPARATION and SHORT-SEPARATION-SINGLE implement the inner-loop interface,

with probability at least $32(k')^3 e^{-g(\lambda,k')}$ in case of SHORT-SEPARATION and $32e^{-g(\lambda,k')}$ in case of SHORT-SEPARATION-SINGLE, it returns a pair $(A', \mathcal{P}')$ that is compatible with Z' in G'.

We verify that the edges added to A for A' are compatible with Z. The connected components of $G[W_{a-1,b+1}] - Z$ are the same as those of $G' - Z'$ except that the component of s' has W_{a-1} in place of s', and the component of t' contains W_{b+1} instead of t' (respectively, are identical but are missing s' and t' if $a = 0$ and/or $b = q + 1$). Thus, the only edges in A that could, in principle, be incompatible with Z are those that were added in place of edges in A' that are incident with s' or t'. But in all cases, the endpoint replacing s' respectively t' is contained in $R_s(Z)$ respectively $R_t(Z)$, implying that they are compatible with Z in G if they are compatible with Z' in G'.

For the family of paths $\mathcal{P}$, note that if $(A', \mathcal{P}')$ is compatible with Z', then for every $P' \in \mathcal{P}'$, the path P' intersects Z' in precisely one edge and that edge belongs to $Z'_{s,t} = Z_{s,t}$. Hence, by appending P' to a path $P \in \mathcal{P}$ we add one intersection of P with Z and that intersection belongs to $Z_{s,t}$. Since there is only one strongly affected stretch and in all other cases the edges appended to the paths of $\mathcal{P}$ are disjoint with Z, $\mathcal{P}$ is a witnessing flow for Z in $G + A$ as desired.

Furthermore, the existence of $\mathcal{P}$ implies that $\lambda_{G+A}(s,t) \geq |\mathcal{P}| = \lambda^*$.

In summary, SAMPLE produces a pair $(A, \mathcal{P})$ that is compatible with Z with probability at least (assuming $c_1 \geq 5$):

- if $a = b$ and $k = |Z'|$:
$$2^{-5} \cdot 32 \cdot e^{-g(\lambda,k)} = e^{-g(\lambda,k)};$$
- if $a = b$ and $k > |Z'|$:
$$\begin{aligned} &2^{-5} k^{-1} 2^{-2(k-k')} e^{-g(\lambda,k')} \\ &\quad \geq e^{-5-\ln k - 2(k-k')} e^{c_2(k-k')(1+\ln k)} e^{-g(\lambda,k')} \\ &\quad \geq e^{-g(\lambda,k')}; \end{aligned}$$
- if $a < b$:
$$\begin{aligned} &\frac{1}{16} k^{-3-2(k-k')} \cdot 16(k')^3 e^{-g(\lambda,k')} \\ &\quad \geq e^{-g(\lambda,k)} \cdot k^{c_1(k-k')} \cdot (k')^3 \cdot k^{-3-2(k-k')} \\ &\quad \geq e^{-g(\lambda,k)} \cdot k^{(c_1-2)(k-k')} \cdot (k'/k)^3 \\ &\quad \geq e^{-g(\lambda,k)}. \end{aligned}$$

This finishes the proof of the lemma. ∎

2.4 Cut splits and the inner loop

2.4.1 Single-bundle case

We will now describe an algorithm SHORT-SEPARATION-SINGLE that realizes the first half of the inner-loop interface from the previous section. Given a valid single-bundle instance (G, s, t, k, λ^*) where G decomposes into bundles $W_0 \cup W_1 \cup W_2$, $W_0 = \{s\}$ and $W_2 = \{t\}$, it will run in (probabilistic) polynomial time and always return a λ^*-flow augmenting set A. Moreover, for each (s,t)-cut Z that is valid for (G, s, t, k, λ^*), the set A is compatible with Z with probability at least $32e^{-g(\lambda_G(s,t),k)}$. We call $W_0 = \{s\}$ and $W_2 = \{t\}$ trivial bundles, W_1 is the non-trivial bundle. The algorithm is given in Figure 2.

A few remarks are in place. First, if the algorithm exists at Step 1, then no valid cut Z exists and we can deterministically output a trivially correct answer. Second, sampling of values $(\lambda_1^*, \ldots, \lambda_c^*, k_1, \ldots, k_c)$ does not need to be uniform, but we require that each valid output $(\lambda_1^*, \ldots, \lambda_c^*, k_1, \ldots, k_c)$ is sampled with probability at least k^{-2c}. Note that there are at most k^{2c} valid outputs. This can be achieved by, e.g., sampling each λ_i^* and k_i uniformly at random from $\{1, 2, \ldots, k\}$ and, if the sampled values do not satisfy the requirements, return one fixed partition instead.

Let us now analyse the case when W_1 is connected.

LEMMA 2.6. *Let (G, s, t, k, λ^*) and W_1 be as above, and let Z' be a valid cut for (G, s, t, k, λ^*). If W_1 is a connected bundle, then $\delta(s) \cup \delta(t)$ is a $(\lambda_G(s,t) + 1)$-flow-augmenting set compatible with Z'.*

Proof. Let $A = \delta(s) \cup \delta(t)$. Since Z' is a valid cut, $Z \cap A = \emptyset$ and A is compatible with Z'. Furthermore, if W_1 is a connected bundle, then it consists of a single block. Assume for a contradiction that $G + A$ has an (s,t)-cut C of size $\lambda_G(s,t)$. Then $C \cap A = \emptyset$, and C is an (s,t)-min cut in G disjoint from $\delta(s) \cup \delta(t)$. This contradicts the assumption that W_1 a block. Thus every (s,t)-min cut in G intersects $\delta(s) \cup \delta(t)$ in at least one edge e. Since A contains a copy of e, C is no longer an (s,t)-cut in $G + A$. Hence $G + A$ has no (s,t)-cuts of size $\lambda_G(s,t)$, and $\lambda_{G+A}(s,t) > \lambda_G(s,t)$. ∎

LEMMA 2.7. *Assume that SAMPLE is correct for all inputs $(G', s', t', k', \lambda')$ where either $k' < k$ or $k' = k$ but $\lambda_{G'}(s', t') > \lambda_G(s,t)$, with a success probability of at least $e^{-g(\lambda_{G'}(s',t'),k')}$ for any eligible (s,t)-cut Z. Then SHORT-SEPARATION-SINGLE(G, s, t, k, λ^*) is correct, with a success probability of at least $32e^{-g(\lambda_G(s,t),k)}$.*

Proof. Assume that (G, s, t, k, λ^*) is a valid input. As discussed, we can assume $\lambda_G(s,t) \leq \lambda^* \leq k$. Let Z be a valid cut. If W_1 is a connected bundle, then $A = \delta(s) \cup \delta(t)$

is flow-augmenting and compatible with Z by Lemma 2.6. For the success probability bound, the statement is trivial if $\lambda_{G+A}(s,t) > \lambda^*$ (there is no such Z in this case). Otherwise note that $\lambda_{G+A}(s,t) > \lambda_G(s,t)$ so

$$g(\lambda_G(s,t),k) > g(\lambda_{G+A}(s,t),k) + c_2.$$

Hence, the probability bound follows as long as $e^{c_2} \geq 32$.

If W_1 is a disconnected bundle, let $W_1 = W_1^{(1)} \cup \ldots \cup W_1^{(c)}$ be as in the algorithm. For $i \in [c]$, let $\lambda_i^* = |Z_{s,t} \cap E(W_1^{(i)})|$ and $k_i = |Z \cap E(W_1^{(i)})|$; then by assumption $\lambda^* = \lambda_1^* + \ldots + \lambda_c^*$, $k = k_1 + \ldots k_c$, and $\lambda_i \leq \lambda_i^* \leq k_i$. We note that the algorithm guesses the correct values of k_i and λ_i^* with probability at least k^{-2c}.

Consider some $i \in [c]$ and let $G^{(i)} = G[W_1^{(i)} \cup \{s,t\}]$. Let $Z^{(i)} = Z \cap E(G^{(i)})$, and note that $Z^{(i)}$ is an (s,t)-cut in $G^{(i)}$, with endpoints in different connected components of $G^{(i)} - Z^{(i)}$, and with $Z^{(i)} \cap (\delta(s) \cup \delta(t)) = \emptyset$. Thus $Z^{(i)}$ is eligible for $G^{(i)}$. Furthermore by assumption $|Z_{s,t}^{(i)}| = \lambda_i^*$ and $|Z^{(i)}| = k_i < k$. Thus each call to SAMPLE $(G', s, t, k_i, \lambda_i^*)$ will by assumption return a set A_i such that $\lambda_{G+A_i}(s,t) \geq \lambda_i^*$; since $E(G)$ are partitioned across the instances $G^{(i)}$, it follows that $A = A_1 \cup \ldots \cup A_c$ is a flow-augmenting set with $\lambda_{G+A}(s,t) \geq \lambda^*$. Furthermore, for every $i \in [c]$, with probability at least $e^{-g(\lambda_i,k_i)}$ the set A_i is compatible with $Z^{(i)}$. Now $(A, \mathcal{P})$ is compatible with Z if every pair $(A_i, \mathcal{P}_i)$ is compatible with the respective set $Z^{(i)}$. Hence, the success probability is lower bounded by:

$$\begin{aligned}
&k^{-2c} \cdot \prod_{i=1}^{c} e^{-g(\lambda_i,k_i)} \\
&= \exp\left(-2c\ln k - \sum_{i=1}^{c} (c_1(2k_i - \lambda_i) - c_2)\,(1 + \ln k_i)\right) \\
&\geq \exp\left(-2c\ln k - (1+\ln k)\sum_{i=1}^{c} c_1(2k_i - \lambda_i) - c_2\right) \\
&= \exp(-2c\ln k + (1+\ln k)\,(c_1(2k-\lambda) - c_2) \\
&\qquad + (1+\ln k)(c-1)c_2) \\
&\geq 32e^{-g(\lambda,k)} \cdot \exp(((c-1)c_2 - 2c)\ln k \\
&\qquad + ((c-1)c_2 - \ln 32)) \\
&\geq 32e^{-g(\lambda,k)}.
\end{aligned}$$

In the above we have used that $c_1 > c_2$ and, in the last inequality, that $c \geq 2$, $c_2 \geq 4 > \ln 32$. This finishes the proof of the lemma. □

2.4.2 Multiple-bundle case We will now describe an algorithm SHORT-SEPARATION that realizes the inner-loop interface from the previous section. Given a valid multiple-bundle instance (G, s, t, k, λ^*) where G decomposes into bundles $W_0 \cup \ldots \cup W_{q+1}$, with $2 \leq q \leq 2k$, and $W_0 = \{s\}$ and $W_{q+1} = \{t\}$, and with $\lambda := \lambda_G(s,t) < \lambda^*$ it will run in (probabilistic) polynomial time and always return an (s,t)-flow augmenting set A. Moreover, for each (s,t)-cut Z that is valid for (G, s, t, k, λ^*), the set A is compatible with Z with probability at least $32k^3 e^{-g(\lambda_G(s,t),k)}$. We call $W_0 = \{s\}$ and $W_{q+1} = \{t\}$ trivial bundles; all others are called non-trivial bundles.

The algorithm is shown in Figure 3, but to discuss it we need a few results. Assume that Z is a λ^*-eligible (s,t)-cut which affects every non-trivial bundle $W_1, \ldots, W_q$ of G. Let C be the min-cut between W_1 and W_2. We define a *cut labelling* $\varphi_Z \colon V(C) \to \{s, t, \perp\}$ of C by Z as

$$\varphi_Z(v) = \begin{cases} s & v \in R_s(Z) \\ t & v \in R_t(Z) \\ \perp & \text{otherwise.} \end{cases}$$

For every edge $uv \in C$ with $u \in V(W_1)$ and $v \in V(W_2)$, the *type* of the edge uv is the pair $\varphi_Z(uv) := (\varphi_Z(u), \varphi_Z(v))$. Let $\Gamma = \{s, t, \perp\} \times \{s, t, \perp\}$ be the set of types. For a type $\gamma \in \Gamma$, let λ_γ be the number of edges $e \in C$ with $\varphi_Z(e) = \gamma$. The types (s,s) and (t,t) are somewhat special; we denote $\Gamma_0 = \Gamma \setminus \{(s,s),(t,t)\}$ and $\lambda_0 = \sum_{\gamma \in \Gamma_0} \lambda_\gamma$. Furthermore, let $\lambda_\leftarrow = \{t, \perp\} \times \{s, t, \perp\}$ and $\lambda_\rightarrow = \{s, t, \perp\} \times \{s, \perp\}$.

Let $Z_1 = Z \cap E(W_1)$, $Z_2 = Z \cap E(W_{2,q})$ and $Z_C = Z \cap C$. Note that $Z = Z_1 \cup Z_2 \cup Z_C$ is a partition of Z. We make some simple observations.

PROPOSITION 2.10. *The following hold.*

1. $Z_{s,t} \cap C = \{e \in C \mid \phi_Z(e) \in \{(s,t),(t,s)\}\}$;
2. *If* $Z_{s,t} \cap E(W_1) \neq \emptyset$, *then there exists* $u \in V(W_1) \cap V(C)$ *with* $\varphi_Z(u) = t$;
3. *If* $Z_{s,t} \cap E(W_{2,q}) \neq \emptyset$, *then there exists* $v \in V(W_2) \cap V(C)$ *with* $\varphi_Z(v) = s$;
4. *For every* $uv \in C$ *such that* $\varphi_Z(u) \neq \varphi_Z(v)$, *we have* $uv \in Z_C$. *Conversely, if* $uv \in Z_C$, *then* $\varphi_Z(u) \neq \varphi_Z(v)$ *or* $\varphi_Z(u) = \varphi_Z(v) = \perp$.
5. $Z_1 \cup Z_C \neq \emptyset$ *and* $Z_2 \cup Z_C \neq \emptyset$.
6. $|Z_1 \cup Z_C| \geq \lambda_0 + \lambda_{(t,t)}$ *and* $|Z_2 \cup Z_C| \geq \lambda_0 + \lambda_{(s,s)}$.
7. $|Z_1| \geq \lambda_\leftarrow$ *and* $|Z_2| \geq \lambda_\rightarrow$.

We use this to show the correctness of the algorithm.

LEMMA 2.8. *Assume that* SAMPLE *is correct for all inputs* $(G', s', t', k', \lambda')$ *where either* $k' < k$ *or* $k' = k$ *but* $(k' - \lambda_{G'}(s',t')) < (k - \lambda_G(s,t))$, *with a success probability of at least* $e^{-g(\lambda_{G'}(s',t'),k')}$. *Then* SHORT-SEPARATION(G, s, t, k, λ^*) *is correct, with a success probability of at least* $32k^3 e^{-g(\lambda_G(s,t),k)}$.

Algorithm SHORT-SEPARATION-SINGLE(G, s, t, k, λ^*)

1. If (G, s, t, k, λ^*) is not a valid input, or it does not hold that $\lambda_G(s,t) \le \lambda^* \le k$, then set A to be $\max(k+1, \lambda^*)$ copies of $\{s,t\}$, $\mathcal{P}$ to be any λ^* of these copies, and return $(A, \mathcal{P})$.
2. Let $V = W_0 \cup W_1 \cup W_2$ be the partition of G into bundles.
3. If W_1 is a connected bundle:
   (a) Let $A_0 = \delta(s) \cup \delta(t)$.
   (b) Compute $(A, \mathcal{P}) \leftarrow$SAMPLE$(G + A_0, s, t, k, \lambda^*)$.
   (c) Return $(A_0 \cup A, \mathcal{P})$.
4. Otherwise:
   (a) Let $W_1 = W_1^{(1)} \cup \ldots \cup W_1^{(c)}$ be the partition of $G[W_1]$ into connected components, and for each $i \in [c]$ let λ_i be the amount of (s,t)-flow routed through $W_1^{(c)}$; i.e., $\lambda = \lambda_1 + \ldots + \lambda_c$ where $\lambda_i > 0$ for each $i \in [c]$
   (b) Randomly sample partitions $\lambda^* = \lambda_1^* + \ldots + \lambda_c^*$ and $k = k_1 + \ldots k_c$ such that $\lambda_i \le \lambda_i^* \le k_i$ for each $i \in [c]$.
   (c) For every $i \in [c]$, let $G^{(i)} = G[W_1^{(i)} \cup \{s,t\}]$ and compute $(A_i, \mathcal{P}_i) \leftarrow$SAMPLE$(G^{(i)}, s, t, k_i, \lambda_i^*)$.
   (d) Return $(A := \bigcup_{i=1}^c A_i, \mathcal{P} := \bigcup_{i=1}^c \mathcal{P}_i)$.

Figure 2: Inner loop: Algorithm for a single bundle

Proof. First observe that if a call $(G_i, s, t, k_i, \lambda_i^*)$ is made to SAMPLE, then s and t are connected in G_i. Indeed, $G[W_1 \cup \{s\}]$ is connected, and if $\varphi^{-1}(t) \cap V(W_1) = \emptyset$ then the algorithm always guesses $\lambda_1^* = 0$, hence no recursive call is made. Similarly, $G[W_{2,q} \cup \{t\}]$ is connected and if s is not adjacent to $V(W_2)$ in $G + A_0$ then no recursive call into G_2 is made. Hence each recursive call is only made to a connected graph G_i and we can assume that $\mathcal{P}_i$ is a flow of size λ_i^* in $G_i + A_i$. We show that $\mathcal{P}$ is a flow of size λ^* in $G + A$, which implies that $\lambda_{G+A}(s,t) \ge \lambda^*$. Indeed, the paths of $\mathcal{P}_1 \cup \mathcal{P}_2$ exist in $G + A$ and are pairwise edge-disjoint. Furthermore, for every edge $e \in C$ with $\varphi(e) \in \{(s,t),(t,s)\}$, the constucted path $P_e \in \mathcal{P}_C$ is a path from s to t disjoint from $\mathcal{P}_1 \cup \mathcal{P}_2$. Since $|\mathcal{P}_C| = \lambda_C^*$ and $\lambda^* = \lambda_1^* + \lambda_C^* + \lambda_2^*$, $\mathcal{P}$ is as desired.

Next, we consider the probability that $(A, \mathcal{P})$ is compatible with Z. The algorithm correctly guesses (in every bullet, we condition on the previous guesses being correct):

- values λ_γ for $\gamma \in \Gamma$ with probability at least $(1+\lambda)^{-|\Gamma|} \ge k^{-9}$;
- $\varphi = \varphi_Z$ with probability

$$\prod_{e \in C} \frac{\lambda_{\varphi_Z(e)}}{\lambda} = \prod_{\gamma \in \Gamma} \left(\frac{\lambda_\gamma}{\lambda}\right)^{\lambda_\gamma} = \exp\left(-\sum_{\gamma \in \Gamma} \lambda_\gamma \ln(\lambda/\lambda_\gamma)\right).$$

- values $\lambda_1^* = |Z_{s,t} \cap E(W_1)|$ and $\lambda_2^* = |Z_{s,t} \cap E(W_{2,q})|$ with probability at least k^{-2};
- values $k_1 = |Z \cap E(W_1)|$, $k_2 = |Z \cap E(W_2)|$, $k_C = |Z \cap C|$ with probability at least k^{-2}, as there are at most k^2 possible values of (k_1, k_2).

Proposition 2.10 ensures that in all of the above guesses, the correct value of is among one of the options with positive probability. Furthermore, $\lambda_C^* = |Z_{s,t} \cap C|$ is computed (deterministically) by the algorithm.

It was argued above that each recursive call on a graph G_i, $i = 1, 2$, is made only if G_i is connected. We claim that furthermore $Z_1 := Z \cap E(W_1)$ is an eligible (s,t)-cut in G_1. Indeed, $Z_1 \cap \delta(s) = \emptyset$ by assumption, and $Z_1 \cap \delta(t) = \emptyset$ since all edges of $\delta(t)$ in G_1 are from A_0. Furthermore, by assumption, for every vertex u of $N_{G_1}(s)$ and every vertex v of $N_{G_1}(t)$, we have $u \in R_s(Z)$ and $v \in R_t(Z)$. Hence Z_1 in particular cuts every path from u to v in $G[W_1]$, and by cutting all these paths Z_1 must cut s from t in G_1. Finally, no edge of Z_1 goes within a connected component of $G_1 - Z_1$, since the only paths that are added to $G[W_1]$ go between vertices of the same component (either $R_s(Z)$ or $R_t(Z)$) in $G - Z$. Hence with probability at least $e^{-g(\lambda_{G_1}(s,t),k_1)}$ (or 1 if $\lambda_1^* = 0$) the pair $(A_1, \mathcal{P}_1)$ is compatible with Z_1. All these arguments can also be made symmetrically to argue that with probability at least $e^{-g(\lambda_{G_2}(s,t),k_2)}$ (or 1 if $\lambda_2^* = 0$), $(A_2, \mathcal{P}_2)$ is compatible with Z_2.

By assumption, A_{st} is compatible with Z. Also, if $\varphi = \varphi_Z$, then every path $P \in \mathcal{P}_C$ intersects Z in exactly one edge and this edge belongs to $Z_{s,t}$.

It remains to wrap up the proof of the bound the probability that $(A = A_{st} \cup A_1 \cup A_2, \mathcal{P} = \mathcal{P}_1 \cup \mathcal{P}_2 \cup \mathcal{P}_C)$ is compatible with Z. First, consider a corner case when $\lambda_{(s,s)} = \lambda$, that is, φ_Z is constant at (s,s). Then $k_1 \ge 1$, $k_2 \le k-1$, $k_C = 0$, $\lambda_{G_2}(s,t) \ge \lambda_G(s,t)$, and the recursive call on G_1 is not made. Furthermore, once $\lambda_{(s,s)} = \lambda$ is guessed, φ is defined deterministically. Hence, for sufficiently large constant c_1, $(A, \mathcal{P})$ is compatible with Z

Algorithm SHORT-SEPARATION(G, s, t, k, λ^*)

1. If (G, s, t, k, λ^*) is not a valid multiple-bundle input, then return $k+1$ copies of the edge $\{s, t\}$ and stop.
2. Let $V = W_0 \cup \ldots \cup W_{q+1}$ be the partition of G into bundles. Let C be the min-cut between W_1 and W_2.
3. Randomly sample values $0 \le \lambda_\gamma \le \lambda$ for $\gamma \in \Gamma$ such that $\sum_{\gamma\in\Gamma} \lambda_\gamma = \lambda = |C|$. Denote $\lambda_0 = \sum_{\gamma\in\Gamma_0} \lambda_\gamma$.
4. For every edge $uv \in C$ with $u \in V(W_1)$ and $v \in V(W_2)$, guess a label $\varphi(uv) \in \Gamma$ with the probability of $\varphi(uv) = \gamma$ being λ_γ/λ.
   (a) Define $\varphi(u)$ and $\varphi(v)$ such that $(\varphi(u), \varphi(v)) = \varphi(uv)$. If a vertex x obtains two distinct values $\varphi(x)$ in this process, return A being $k+1$ edges st and stop.
   (b) Let λ^*_C be the number of edges $e \in C$ such that $\varphi(e) \in \{(s,t),(t,s)\}$.
5. Let A_{st} contain $k+1$ copies of each edge $\{u, v\}$ with $u, v \in \{s\} \cup \varphi^{-1}(s)$ or with $u, v \in \{t\} \cup \varphi^{-1}(t)$
6. Compute a set $\mathcal{P}_C$ of size λ^*_C as follows: for every $e \in C$ such that $\varphi(e) \in \{(s,t),(t,s)\}$, let $e = uv$ be such that $\varphi(u) = s$ and $\varphi(v) = t$,and add to $\mathcal{P}_C$ a three-edge path P_e consisting of the edges $su \in A_{st}$, e, and $tv \in A_{st}$.
7. Randomly sample a partition $\lambda^* = \lambda^*_1 + \lambda^*_C + \lambda^*_2$ subject to the following constraints:
   (a) $\lambda^*_1 \ge \lambda_{(t,s)} + \lambda_{(t,t)} + \lambda_{(t,\perp)}$ and $\lambda^*_1 = 0$ if $\lambda_{(t,s)} = \lambda_{(t,t)} = \lambda_{(t,\perp)} = 0$.
   (b) $\lambda^*_2 \ge \lambda_{(s,s)} + \lambda_{(t,s)} + \lambda_{(\perp,s)}$ and $\lambda^*_2 = 0$ if $\lambda_{(s,s)} = \lambda_{(t,s)} = \lambda_{(\perp,s)} = 0$.
8. Randomly sample a partition $k = k_1 + k_C + k_2$ subject to the following constraints:
   (a) $\lambda^*_1 \le k_1$, $\lambda_0 + \lambda_{(t,t)} \le k_1 + k_C$, $1 \le k_1$, $\lambda_\leftarrow \le k_1$;
   (b) $\lambda^*_2 \le k_2$, $\lambda_0 + \lambda_{(s,s)} \le k_2 + k_C$, $1 \le k_2$, $\lambda_\rightarrow \le k_2$;
   (c) $\sum_{\gamma\in\Gamma_0\setminus\{(\perp,\perp)\}} \lambda_\gamma \le k_C \le \sum_{\gamma\in\Gamma_0} \lambda_\gamma$.
9. Construct a flow-augmenting set A_1 and a flow in W_1:
   (a) Let $G_1 = (G + A_{st})[W_1 \cup \{s, t\}]$;
   (b) Compute $(A_1, \mathcal{P}_1) \leftarrow$ SAMPLE$(G_1, s, t, k_1, \lambda^*_1)$.
10. Construct a flow-augmenting set A_2 in $W_{2,q}$:
    (a) Let $G_2 = (G + A_{st})[W_{2,q} \cup \{s, t\}]$.
    (b) Compute $(A_2, \mathcal{P}_2) \leftarrow$ SAMPLE$(G_2, s, t, k_2, \lambda^*_2)$.
11. Return $(A = A_{st} \cup A_1 \cup A_2, \mathcal{P} = \mathcal{P}_C \cup \mathcal{P}_1 \cup \mathcal{P}_2)$.

Figure 3: The inner loop algorithm for multiple-bundle case.

with probability at least

$$\begin{aligned} k^{-13} e^{-g(\lambda_{G_2}(s,t),k_2)} &\ge k^{-13} e^{-g(\lambda,k-1)} \\ &\ge \exp\left(-16\ln k - \ln 16 + c_2(1+\ln 4)\right) 16k^3 e^{-g(\lambda,k)} \\ &\ge e^{-g(\lambda,k)}. \end{aligned}$$

A symmetric argument holds if $\lambda_{(t,t)} = \lambda$, that is, φ_Z is constant at (t, t).

For the general case, observe that even if the recursive call on G_i is not invoked due to $\lambda^*_i = 0$, then $k_i \ge 1$ and $\lambda_{G_i}(s,t) \le k_i$ so $e^{-g(\lambda_{G_i}(s,t),k_i)} \le 1$. Thus, we can use $e^{-g(\lambda_{G_i}(s,t),k_i)}$ as a lower bound on the success probability of the recursive call regardless of whether it was actually invoked.

By the above discussion, the probability that A is compatible with Z is at least

$$k^{-13} \cdot \exp\left(-\sum_{\gamma\in\Gamma} \lambda_\gamma \ln(\lambda/\lambda_\gamma)\right) \cdot e^{-g(\lambda_{G_1}(s,t),k_1)} e^{-g(\lambda_{G_2}(s,t),k_2)}. \tag{2.1}$$

We start by analysing the second term of the above bound. By the concavity of $\ln(\cdot)$, we have that

$$\sum_{\gamma\in\Gamma_0} \lambda_\gamma \ln \lambda_\gamma \ge \lambda_0 \ln(\lambda_0/|\Gamma_0|) = \lambda_0 \ln \lambda_0 - \lambda_0 \ln 7. \tag{2.2}$$

Hence,

$$\begin{aligned} \sum_{\gamma\in\Gamma} \lambda_\gamma \ln(\lambda/\lambda_\gamma) &\le \lambda_{(s,s)} \ln(\lambda/\lambda_{(s,s)}) \\ &\quad + \lambda_{(t,t)} \ln(\lambda/\lambda_{(t,t)}) + \lambda_0 \ln(\lambda/\lambda_0) + \lambda_0 \ln 7. \end{aligned} \tag{2.3}$$

Denote $x_1 = k_1 - \lambda_0$, $x_2 = k_2 - \lambda_0$, $x_0 = \lambda_0$, and $x = x_1 + x_2 + 2x_0 = k_1 + k_2$. By entropy maximization,

$$\begin{aligned} &\lambda_{(s,s)} \ln(\lambda/\lambda_{(s,s)}) + \lambda_{(t,t)} \ln(\lambda/\lambda_{(t,t)}) + \lambda_0 \ln(\lambda/\lambda_0) \\ &\le \lambda_{(s,s)} \ln(x/x_1) + \lambda_{(t,t)} \ln(x/x_2) + \lambda_0 \ln(x/(2x_0)) \\ &\le x_1 \ln(x/x_1) + x_2 \ln(x/x_2) + 2x_0 \ln(x/(2x_0)). \end{aligned} \tag{2.4}$$

We also need the following observation:

CLAIM 1. *It holds that*

$$k_1 - \lambda_{G_1}(s,t) + k_2 - \lambda_{G_2}(s,t) \le k - \lambda_G(s,t) + \lambda_{(\perp,\perp)}.$$

Proof. From Proposition 2.10(4.), we infer that

$$|C| - k_C - \lambda_{(\bot,\bot)} \leq \lambda_{(s,s)} + \lambda_{(t,t)}.$$

Since in G_1, an endpoint of every edge $e \in C$ with $\varphi_Z(e) = (t,t)$ is connected to t with $k+1$ edges, we have $\lambda_{G_1}(s,t) \geq \lambda_{(t,t)}$. Symmetrically, $\lambda_{G_2}(s,t) \geq \lambda_{(s,s)}$. As $k_1 + k_2 + k_C = k$ and $\lambda_G(s,t) = |C|$, the claim follows. □

To wrap up the analysis, we need the following property of the $z \mapsto z \ln z$ function (for completeness, we provide a proof in [18]):

Claim 2. *Let $f(z) = z \ln z$ for $z > 0$. For every constant $C_1 > 0$ there exists a constant $C_2 > 0$ such that for every $x_1, x_2, x_0 > 0$ it holds that*

$$\begin{aligned} C_2 f(x_1 + x_2 + 2x_0) + f(x_1) + f(x_2) + f(2x_0) \\ \geq f(x_1 + x_2 + 2x_0) + C_2 f(x_1 + x_0) \\ + C_2 f(x_2 + x_0) + C_1 x_0. \end{aligned}$$

Claim 2 for $C_1 = c_2 + \ln 7$ implies an existence of $C_2 > 0$ (depending on c_2) such that

(2.5)
$$\begin{aligned} x_1 \ln(x/x_1) + x_2 \ln(x/x_2) + 2x_0 \ln(x/(2x_0)) \\ + x_0(c_2 + \ln 7) \leq C_2 \left(x \ln x - k_1 \ln k_1 - k_2 \ln k_2\right). \end{aligned}$$

Using the definition of $g(\cdot,\cdot)$, the fact that $\lambda_{(\bot,\bot)} \leq x_0$, and Claim 1, we obtain that

(2.6)
$$\begin{aligned} g(\lambda_G(s,t), k) \geq g(\lambda_{G_1}(s,t), k_1) + g(\lambda_{G_2}(s,t), k_2) \\ + c_1 \left(x \ln x - k_1 \ln k_1 - k_2 \ln k_2\right) \\ + c_2(1 + \ln k) - c_2 x_0. \end{aligned}$$

Thus, we bound the negated exponent of the probability bound of (2.1) as follows:

$$\begin{aligned} & 13 \ln k + \sum_{\gamma \in \Gamma} \lambda_\gamma \ln(\lambda/\lambda_\gamma) + g(\lambda_{G_1}(s,t), k_1) \\ & \quad + g(\lambda_{G_2}(s,t), k_2) \\ & \qquad \text{by (2.3) and (2.4)} \\ & \leq 13 \ln k + x_1 \ln(x/x_1) + x_2 \ln(x/x_2) \\ & \quad + 2x_0 \ln(x/(2x_0)) + x_0 \ln 7 \\ & \quad + g(\lambda_{G_1}(s,t), k_1) + g(\lambda_{G_2}(s,t), k_2) \\ & \qquad \text{by (2.6)} \\ & \leq 13 \ln k + x_1 \ln(x/x_1) + x_2 \ln(x/x_2) \\ & \quad + 2x_0 \ln(x/(2x_0)) + x_0 \ln 7 \\ & \quad + g(\lambda_G(s,t), k) + c_2 x_0 - c_2(1 + \ln k) \\ & \quad + c_1(x_1 + x_0) \ln(x_1 + x_0) \\ & \quad + c_1(x_2 + x_0) \ln(x_2 + x_0) - c_1 x \ln x \\ & \qquad \text{by (2.5)}, c_2 \geq 16,\ c_1 \geq C_2 \\ & \leq g(\lambda_G(s,t), k) - 3 \ln k - \ln 32 \end{aligned}$$

This finishes the proof of the lemma. □

Due to space restrictions, in this extended abstract we omit the details of the implementation of the algorithm in time $k^{\mathcal{O}(1)}\mathcal{O}(m)$. Other than that, Theorem 1.1 follows from Lemma 2.5, Lemma 2.7 and Lemma 2.8.

3 Conclusions

We would like to conclude with conjecturing an existence of a flow-augmentation technique in directed graphs, at least restricted to minimal (s,t)-cuts. More formally, we propose the following:

Conjecture. There exists a randomized fixed-parameter algorithm that, given a directed multigraph G with two designated vertices $s, t \in V(G)$ and a parameter k, samples a multiset A of arcs such that the size of a maximum (s,t)-flow in $G + A$ is strictly larger than in G and for every minimal (s,t)-cut Z of size at most k that is not a minimum (s,t)-cut, Z remains an (s,t)-cut in $G + A$ with probability bounded from below by $1/f(k)$ for a computable function f.

As discussed in the introduction, a positive resolution of the above conjecture would lead to a (randomized) fixed-parameter algorithm for Bi-objective (s,t)-cut and the notorious ℓ-Chain SAT problem. Furthermore, techniques used for proving the conjecture may be helpful in proving tractability of Directed Multicut for three terminal pairs [28].

We also note several further directions of inquiry regarding the parameterized complexity of Min SAT(Γ) and more general optimization CSP problems, e.g., valued CSPs [33, 19]. The immediate question is to extend the Min SAT(Γ) complexity characterization to general finite Boolean languages, including $(u \to v)$ constraints. Some challenges here, beyond ℓ-Chain SAT, include directed versions of Coupled Min-Cut, or even more generally bijunctive languages Γ (i.e., with relations expressible via 2-CNF formulas) where for each $R \in \Gamma$, the constraint graph H_R as defined in [18, Section 6] is $2K_2$-free.

More ambitiously, the question can be broadened from Min SAT to more general Valued CSP problems (Boolean or otherwise). Here, the parameter can either be

taken to be the solution cost, for integer-valued languages, or the number of falsified constraints in an optimal solution.

References

[1] É. Bonnet, L. Egri, B. Lin, and D. Marx. Fixed-parameter approximability of boolean mincsps. *CoRR*, abs/1601.04935v2, 2018.

[2] N. Bousquet, J. Daligault, and S. Thomassé. Multicut is FPT. *SIAM J. Comput.*, 47(1):166–207, 2018.

[3] R. Chitnis, M. Cygan, M. Hajiaghayi, M. Pilipczuk, and M. Pilipczuk. Designing FPT algorithms for cut problems using randomized contractions. *SIAM J. Comput.*, 45(4):1171–1229, 2016.

[4] R. Chitnis, L. Egri, and D. Marx. List H-coloring a graph by removing few vertices. *Algorithmica*, 78(1):110–146, 2017.

[5] R. H. Chitnis, M. Hajiaghayi, and D. Marx. Fixed-parameter tractability of directed multiway cut parameterized by the size of the cutset. *SIAM J. Comput.*, 42(4):1674–1696, 2013.

[6] N. Creignou and H. Vollmer. Boolean constraint satisfaction problems: When does Post's lattice help? In N. Creignou, P. G. Kolaitis, and H. Vollmer, editors, *Complexity of Constraints - An Overview of Current Research Themes [Result of a Dagstuhl Seminar]*, volume 5250 of *Lecture Notes in Computer Science*, pages 3–37. Springer, 2008.

[7] M. Cygan, F. V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. *Parameterized Algorithms*. Springer, 2015.

[8] M. Cygan, P. Komosa, D. Lokshtanov, M. Pilipczuk, M. Pilipczuk, S. Saurabh, and M. Wahlström. Randomized contractions meet lean decompositions. *CoRR*, abs/1810.06864, 2018.

[9] M. Cygan, D. Lokshtanov, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. Minimum bisection is fixed-parameter tractable. *SIAM J. Comput.*, 48(2):417–450, 2019.

[10] M. Cygan, M. Pilipczuk, M. Pilipczuk, and J. O. Wojtaszczyk. On multiway cut parameterized above lower bounds. *ACM Trans. Comput. Theory*, 5(1):3:1–3:11, 2013.

[11] R. Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012.

[12] J. Guo, J. Gramm, F. Hüffner, R. Niedermeier, and S. Wernicke. Compression-based fixed-parameter algorithms for feedback vertex set and edge bipartization. *J. Comput. Syst. Sci.*, 72(8):1386–1396, 2006.

[13] Y. Iwata, K. Oka, and Y. Yoshida. Linear-time FPT algorithms via network flow. In *SODA*, pages 1749–1761. SIAM, 2014.

[14] Y. Iwata, M. Wahlström, and Y. Yoshida. Half-integrality, LP-branching, and FPT algorithms. *SIAM J. Comput.*, 45(4):1377–1411, 2016.

[15] Y. Iwata, Y. Yamaguchi, and Y. Yoshida. 0/1/all CSPs, half-integral A-path packing, and linear-time FPT algorithms. In *FOCS*, pages 462–473. IEEE Computer Society, 2018.

[16] K. Kawarabayashi and M. Thorup. The minimum k-way cut of bounded size is fixed-parameter tractable. In *FOCS*, pages 160–169. IEEE Computer Society, 2011.

[17] S. Khanna, M. Sudan, L. Trevisan, and D. P. Williamson. The approximability of constraint satisfaction problems. *SIAM J. Comput.*, 30(6):1863–1920, 2000.

[18] E. J. Kim, S. Kratsch, M. Pilipczuk, and M. Wahlström. Solving hard cut problems via flow-augmentation. *CoRR*, abs/2007.09018, 2020.

[19] V. Kolmogorov, A. A. Krokhin, and M. Rolínek. The complexity of general-valued CSPs. *SIAM J. Comput.*, 46(3):1087–1110, 2017.

[20] S. Kratsch, S. Li, D. Marx, M. Pilipczuk, and M. Wahlström. Multi-budgeted directed cuts. In *IPEC*, volume 115 of *LIPIcs*, pages 18:1–18:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.

[21] D. Lokshtanov and D. Marx. Clustering with local restrictions. *Inf. Comput.*, 222:278–292, 2013.

[22] D. Lokshtanov, N. S. Narayanaswamy, V. Raman, M. S. Ramanujan, and S. Saurabh. Faster parameterized algorithms using linear programming. *ACM Trans. Algorithms*, 11(2):15:1–15:31, 2014.

[23] D. Marx. Parameterized graph separation problems. *Theor. Comput. Sci.*, 351(3):394–406, 2006.

[24] D. Marx, B. O'Sullivan, and I. Razgon. Finding small separators in linear time via treewidth reduction. *ACM Trans. Algorithms*, 9(4):30:1–30:35, 2013.

[25] D. Marx and I. Razgon. Constant ratio fixed-parameter approximation of the edge multicut problem. *Inf. Process. Lett.*, 109(20):1161–1166, 2009.

[26] D. Marx and I. Razgon. Fixed-parameter tractability of multicut parameterized by the size of the cutset. *SIAM J. Comput.*, 43(2):355–388, 2014.

[27] C. H. Papadimitriou and M. Yannakakis. Multiobjective query optimization. In *PODS*. ACM, 2001.

[28] M. Pilipczuk and M. Wahlström. Directed multicut is $W[1]$-hard, even for four terminal pairs. *ACM Trans. Comput. Theory*, 10(3):13:1–13:18, 2018.

[29] E. L. Post. *The Two-Valued Iterative Systems of Mathematical Logic. (AM-5)*. Princeton University Press, 1941.

[30] M. S. Ramanujan and S. Saurabh. Linear time parameterized algorithms via skew-symmetric multicuts. In *SODA*, pages 1739–1748. SIAM, 2014.

[31] I. Razgon and B. O'Sullivan. Almost 2-SAT is fixed-parameter tractable. *J. Comput. Syst. Sci.*, 75(8):435–450, 2009.

[32] B. A. Reed, K. Smith, and A. Vetta. Finding odd cycle transversals. *Oper. Res. Lett.*, 32(4):299–301, 2004.

[33] J. Thapper and S. Zivny. The complexity of finite-valued CSPs. *J. ACM*, 63(4):37:1–37:33, 2016.