Parameterized Algorithms for Graph Partitioning Problems

Hadas Shachnai and Meirav Zehavi

Department of Computer Science, Technion, Haifa 32000, Israel
{hadas,meizeh}@cs.technion.ac.il

Abstract. We study a broad class of graph partitioning problems, where each problem is specified by a graph $G=(V,E)$, and parameters k and p. We seek a subset $U \subseteq V$ of size k, such that $\alpha_1 m_1 + \alpha_2 m_2$ is at most (or at least) p, where $\alpha_1, \alpha_2 \in \mathbb{R}$ are constants defining the problem, and m_1, m_2 are the cardinalities of the edge sets having both endpoints, and exactly one endpoint, in U, respectively. This class of *fixed cardinality graph partitioning problems (FGPP)* encompasses Max $(k,n-k)$-Cut, Min k-Vertex Cover, k-Densest Subgraph, and k-Sparsest Subgraph. Our main result is an $O^*(4^{k+o(k)}\Delta^k)$ algorithm for any problem in this class, where $\Delta \geq 1$ is the maximum degree in the input graph. This resolves an open question posed by Bonnet et al. [IPEC 2013]. We obtain faster algorithms for certain subclasses of FGPPs, parameterized by p, or by $(k+p)$. In particular, we give an $O^*(4^{p+o(p)})$ time algorithm for Max $(k, n-k)$-Cut, thus improving significantly the best known $O^*(p^p)$ time algorithm.

1 Introduction

Graph partitioning problems arise in many areas including VLSI design, data mining, parallel computing, and sparse matrix factorizations (see, e.g., [1, 12, 7]). We study a broad class of graph partitioning problems, where each problem is specified by a graph $G=(V,E)$, and parameters k and p. We seek a subset $U \subseteq V$ of size k, such that $\alpha_1 m_1 + \alpha_2 m_2$ is at most (or at least) p, where $\alpha_1, \alpha_2 \in \mathbb{R}$ are constants defining the problem, and m_1, m_2 are the cardinalities of the edge sets having both endpoints, and exactly one endpoint, in U, respectively. This class encompasses such fundamental problems as Max and Min $(k,n-k)$-Cut, Max and Min k-Vertex Cover, k-Densest Subgraph, and k-Sparsest Subgraph. For example, Max $(k,n-k)$-Cut is a max-FGPP (i.e., maximization FGPP) satisfying $\alpha_1 = 0$ and $\alpha_2 = 1$, Min $(k,n-k)$-Cut is a min-FGPP (i.e., minimization FGPP) satisfying $\alpha_1 = 0$ and $\alpha_2 = 1$, and Min k-Vertex Cover is a min-FGPP satisfying $\alpha_1 = \alpha_2 = 1$.

A parameterized algorithm with parameter k has running time $O^*(f(k))$ for some function f, where O^* hides factors polynomial in the input size. In this paper, we develop a parameterized algorithm with parameter $(k+\Delta)$ for the class of all FGPPs, where $\Delta \geq 1$ is the maximum degree in the graph G. For certain subclasses of FGPPs, we develop algorithms parameterized by p, or by $(k+p)$.

Related Work: Parameterized by k, Max and Min $(k,n-k)$-Cut, and Max and Min k-Vertex Cover are W[1]-hard [8, 4, 11]. Moreover, k-Clique and k-Independent Set, two well-known W[1]-hard problems [9], are special cases of k-Densest Subgraph where $p=k(k-1)$, and k-Sparsest Subgraph where $p=0$, respectively. Therefore, parameterized by $(k+p)$, k-Densest Subgraph and k-Sparsest Subgraph are W[1]-hard. Cai et al. [5] and Bonnet et al. [2] studied the parameterized complexity of FGPPs with respect to $(k+\Delta)$. Cai et al. [5] gave $O^*(2^{(k+1)\Delta})$ time algorithms for k-Densest Subgraph and k-Sparsest Subgraph. Recently, Bonnet et al. [2] presented an $O^*(\Delta^k)$ time algorithm for *degrading* FGPPs. This subclass includes max-FGPPs in which $\alpha_1/2 \leq \alpha_2$, and min-FGPPs in which $\alpha_1/2 \geq \alpha_2$.[1] They also proposed an $O^*(k^{2k}\Delta^{2k})$ time algorithm for all FGPPs, and posed as an open question the existence of constants a and b such that any FGPP can be solved in time $O^*(a^k\Delta^{bk})$. In this paper we answer this question affirmatively, by developing an $O^*(4^{k+o(k)}\Delta^k)$ time algorithm for any FGPP.

Parameterized by p, Max and Min k-Vertex Cover can be solved in times $O^*(1.396^p)$ and $O^*(4^p)$, respectively, and in randomized times $O^*(1.2993^p)$ and $O^*(3^p)$, respectively [14]. Moreover, Max $(k,n-k)$ Cut can be solved in time $O^*(p^p)$ [2], and Min $(k,n-k)$ Cut can be solved in time $O(2^{O(p^3)})$ [6]. Parameterized by $(k+p)$, Min $(k,n-k)$ Cut can be solved in time $O^*(k^{2k}(k+p)^{2k})$ [2].

We note that the parameterized complexity of FGPPs has also been studied with respect to other parameters, such as the treewidth and the vertex cover number of G (see, e.g., [13, 3, 2]).

Contribution: Our main result is an $O^*(4^{k+o(k)}\Delta^k)$ time algorithm for the class of all FGPPs, answering affirmatively the question posed by Bonnet et al. [2] (see Section 2). In Section 3, we develop an $O^*(4^{p+o(p)})$ time algorithm for Max $(k,n-k)$-Cut, that significantly improves the $O^*(p^p)$ running time obtained in [2]. We also obtain (in Section 4) an $O^*(2^{k+\frac{p}{\alpha_2}+o(k+p)})$ time algorithm for the subclass of *positive* min-FGPPs, in which $\alpha_1 \geq 0$ and $\alpha_2 > 0$. Finally, we develop (in Section 5) a faster algorithm for non-degarding positive min-FGPPs (i.e., min-FGPPs satisfying $\alpha_2 \geq \frac{\alpha_1}{2} > 0$). In particular, we thus solve Min k-Vertex Cover in time $O^*(2^{p+o(p)})$, improving the previous *randomized* $O^*(3^p)$ time algorithm.

Techniques: We obtain our main result by establishing an interesting reduction from non-degrading FGPPs to the Weighted k'-Exact Cover (k'-WEC) problem (see Section 2). Building on this reduction, combined with an algorithm for degrading FGPPs given in [2], and an algorithm for k'-WEC given in [18], we develop an algorithm for any FGPP. To improve the running time of our algorithm, we use a fast construction of representative families [10, 17].

In designing algorithms for FGPPs, parameterized by p or $(k+p)$, we use as a key tool *randomized separation* [5] (see Sections 3–5). Roughly speaking, randomized separation finds a 'good' partition of the nodes in the input graph G via randomized coloring of the nodes in *red* or *blue*. If a solution exists, then,

[1] A max-FGPP (min-FGPP) is non-degrading if $\alpha_1/2 \geq \alpha_2$ ($\alpha_1/2 \leq \alpha_2$).

with some positive probability, there is a set X of *only* red nodes that is a solution, such that *all* the neighbors of nodes in X that are outside X are blue. Our algorithm for Max $(k,n-k)$-Cut makes non-standard use of randomized separation, in requiring that only *some* of the neighbors outside X of nodes in X are blue. This yields the desired improvement in the running time of our algorithm.

Our algorithm for non-degrading positive FGPPs is based on a somewhat different application of randomized separation, in which we randomly color *edges* rather than the nodes. If a solution exists, then, with some positive probability, there is a node-set X that is a solution, such that *some* edges between nodes in X are red, and *all* edges between nodes in X and nodes outside X are blue. In particular, we require that the subgraph induced by X, and the subgraph induced by X from which we delete all blue edges, contain the same connected components. We derandomize our algorithms using universal sets [16].

Notation: Given a graph $G=(V,E)$ and a subset $X \subseteq V$, let $E(X)$ denote the set of edges in E having both endpoints in X, and let $E(X,V \setminus X)$ denote the set of edges in E having exactly one endpoint in X. Moreover, given a subset $X \subseteq V$, let $\mathrm{val}(X)=\alpha_1|E(X)|+\alpha_2|E(X,V \setminus X)|$.

2 Solving FGPPs in Time $O^*(4^{k+o(k)}\Delta^k)$

In this section we develop an $O^*(4^{k+o(k)}\Delta^k)$ time algorithm for the class of all FGPPs. We use the following steps. In Section 2.1 we show that any non-degrading FGPP can be reduced to the Weighted k'-Exact Cover (k'-WEC) problem, where $k'=k$. Applying this reduction, we then show (in Section 2.2) how to decrease the size of instances of k'-WEC, by using representative families. Finally, we show (in Section 2.3) how to solve any FGPP by using the results in Sections 2.1 and 2.2, an algorithm for k'-WEC, and an algorithm for degrading FGPPs given in [2].

2.1 From Non-Degrading FGPPs to k'-WEC

We show below that any non-degrading max-FGPP can be reduced to the maximization version of k'-WEC. Given a universe U, a family $\mathcal{S}$ of nonempty subsets of U, a function $w:\mathcal{S}\rightarrow\mathbb{R}$, and parameters $k'\in\mathbb{N}$ and $p'\in\mathbb{R}$, we seek a subfamily $\mathcal{S}'$ of disjoint sets from $\mathcal{S}$ satisfying $|\bigcup\mathcal{S}'|=k'$ whose value, given by $\sum_{S\in\mathcal{S}'} w(S)$, is at least p'. Any non-degrading min-FGPP can be similarly reduced to the minimization version of k'-WEC.

Let Π be a max-FGPP satisfying $\frac{\alpha_1}{2}\geq\alpha_2$. Given an instance $\mathcal{I}=(G=(V,E),k,p)$ of Π, we define an instance $f(\mathcal{I})=(U,\mathcal{S},w,k',p')$ of the maximization version of k'-WEC as follows.

- $U=V$.
- $\mathcal{S}=\bigcup_{i=1}^{k}\mathcal{S}_i$, where $\mathcal{S}_i$ contains the node-set of any connected subgraph of G on exactly i nodes.
- $\forall S\in\mathcal{S}: w(S)=\mathrm{val}(S)$.
- $k'=k$, and $p'=p$.

We illustrate the reduction in Figure 1 (see Appendix A). We first prove that our reduction is valid.

Lemma 1. *$\mathcal{I}$ is a yes-instance iff $f(\mathcal{I})$ is a yes-instance.*

Proof. First, assume there is a subset $X \subseteq V$ of size k satisfying $\mathrm{val}(X) \geq p$. Let $G_1 = (V_1,E_1),\ldots,G_t = (V_t,E_t)$, for some $1 \leq t \leq k$, be the *maximal* connected components in the subgraph of G induced by X. Then, for all $1 \leq \ell \leq t$, $V_\ell \in \mathcal{S}$. Moreover, $\displaystyle\sum_{\ell=1}^{t} |V_\ell| = |X| = k'$, and $\displaystyle\sum_{\ell=1}^{t} w(V_\ell) = \mathrm{val}(X) \geq p'$.

Now, assume there is a subfamily of disjoint sets $\{S_1,\ldots,S_t\} \subseteq \mathcal{S}$, for some $1 \leq t \leq k$, such that $\displaystyle\sum_{\ell=1}^{t} |S_\ell| = k'$ and $\displaystyle\sum_{\ell=1}^{t} w(S_\ell) \geq p'$. Thus, there are connected subgraphs $G_1 = (V_1,E_1),\ldots,G_t = (V_t,E_t)$ of G, such that $V_\ell = S_\ell$, for all $1 \leq \ell \leq t$. Let $X_\ell = \bigcup_{j=\ell}^{t} V_j$, for all $1 \leq \ell \leq t$. Clearly, $|X_1| = k$. Since $\frac{\alpha_1}{2} \geq \alpha_2$, we get that

$$\begin{aligned}
\mathrm{val}(X_1) &= \mathrm{val}(V_1) + \mathrm{val}(X_2) + \alpha_1|E(V_1,X_2)| - 2\alpha_2|E(V_1,X_2)| \\
&\geq \mathrm{val}(V_1) + \mathrm{val}(X_2) \\
&= \mathrm{val}(V_1) + \mathrm{val}(V_2) + \mathrm{val}(X_3) + \alpha_1|E(V_2,X_3)| - 2\alpha_2|E(V_2,X_3)| \\
&\geq \mathrm{val}(V_1) + \mathrm{val}(V_2) + \mathrm{val}(X_3) \\
&\ldots \\
&\geq \sum_{\ell=1}^{t} \mathrm{val}(V_\ell).
\end{aligned}$$

Thus, $\mathrm{val}(X_1) \geq \displaystyle\sum_{\ell=1}^{t} w(V_\ell) \geq p$. □

We now bound the number of connected subgraphs in G.

Lemma 2 ([15]). *There are at most $4^i(\Delta-1)^i|V|$ connected subgraphs of G on at most i nodes, which can be enumerated in time $O(4^i(\Delta-1)^i(|V|+|E|)|V|)$.*

Thus, we have the next result.

Lemma 3. *The instance $f(\mathcal{I})$ can be constructed in time $O(4^k(\Delta-1)^k(|V|+|E|)|V|)$. Moreover, for any $1 \leq i \leq k$, $|\mathcal{S}_i| \leq 4^i(\Delta-1)^i|V|$.*

2.2 Decreasing the Size of Inputs for k'-WEC

In this section we develop a procedure, called Decrease, which decreases the size of an instance $(U,\mathcal{S},w,k',p')$ of k'-WEC. To this end, we find a subfamily $\widehat{\mathcal{S}} \subseteq \mathcal{S}$ that contains "enough" sets from $\mathcal{S}$, and thus enables to replace $\mathcal{S}$ by $\widehat{\mathcal{S}}$ without turning a yes-instance to a no-instance. The following definition captures such a subfamily $\widehat{\mathcal{S}}$.

Definition 1. *Given a universe E, nonnegative integers k and p, a family $\mathcal{S}$ of subsets of size p of E, and a function $w: \mathcal{S} \rightarrow \mathbb{R}$, we say that a subfamily $\widehat{\mathcal{S}} \subseteq \mathcal{S}$* max (min) represents *$\mathcal{S}$ if for any pair of sets $X \in \mathcal{S}$, and $Y \subseteq E \setminus X$ such that $|Y| \leq k-p$, there is a set $\widehat{X} \in \widehat{\mathcal{S}}$ disjoint from Y such that $w(\widehat{X}) \geq w(X)$ ($w(\widehat{X}) \leq w(X)$).*

The following result states that small representative families can be computed efficiently.[2]

Theorem 1 ([17]). *Given a constant* $c\geq 1$*, a universe* E*, nonnegative integers* k *and* p*, a family* $\mathcal{S}$ *of subsets of size* p *of* E*, and a function* $w:\mathcal{S}\rightarrow\mathbb{R}$*, a subfamily* $\widehat{\mathcal{S}}\subseteq\mathcal{S}$ *of size at most* $\frac{(ck)^k}{p^p(ck-p)^{k-p}}2^{o(k)}\log|E|$ *that max (min) represents* $\mathcal{S}$ *can be computed in time* $O(|\mathcal{S}|(ck/(ck-p))^{k-p}2^{o(k)}\log|E|+|\mathcal{S}|\log|\mathcal{S}|)$.

We next consider the maximization version of k'-WEC and max representative families. The minimization version of k'-WEC can be similarly handled by using min representative families. Let $\mathsf{RepAlg}(E,k,p,\mathcal{S},w)$ denote the algorithm in Theorem 1 where $c=2$, and let $\mathcal{S}_i=\{S\in\mathcal{S}: |S|=i\}$, for all $1\leq i\leq k'$.

We now present procedure $\mathsf{Decrease}$ (see the pseudocode below), which replaces each family $\mathcal{S}_i$ by a family $\widehat{\mathcal{S}}_i\subseteq\mathcal{S}_i$ that represents $\mathcal{S}_i$. First, we state that procedure $\mathsf{Decrease}$ is correct (the proof is given in Appendix C).

Procedure $\mathsf{Decrease}(U,\mathcal{S},w,k',p')$

1: **for** $i=1,2,\ldots,k'$ **do** $\widehat{\mathcal{S}}_i \Leftarrow \mathsf{RepAlg}(U,k',i,\mathcal{S}_i,w)$. **end for**
2: $\widehat{\mathcal{S}} \Leftarrow \bigcup_{i=1}^{k}\widehat{\mathcal{S}}_i$.
3: **return** $(U,\widehat{\mathcal{S}},w,k',p')$.

Lemma 4. $(U,\mathcal{S},w,k',p')$ *is a yes-instance iff* $(U,\widehat{\mathcal{S}},w,k',p')$ *is a yes-instance.*

Theorem 1 immediately implies the following result.

Lemma 5. *Procedure* $\mathsf{Decrease}$ *runs in time* $O(\sum_{i=1}^{k'}(|\mathcal{S}_i|(\frac{2k'}{2k'-i})^{k'-i}2^{o(k')}\log|U|$ $+|\mathcal{S}_i|\log|\mathcal{S}_i|))$. *Moreover,* $|\widehat{\mathcal{S}}|\leq\sum_{i=1}^{k'}\frac{(2k')^{k'}}{i^i(2k'-i)^{k'-i}}2^{o(k')}\log|U|\leq 2.5^{k'+o(k')}\log|U|$.

2.3 An Algorithm for Any FGPP

We now present $\mathsf{FGPPAlg}$, which solves any FGPP in time $O^*(4^{k+o(k)}\Delta^k)$. Assume w.l.o.g that $\Delta\geq 2$, and let $\mathsf{DegAlg}(G,k,p)$ denote the algorithm solving any degrading FGPP in time $O((\Delta+1)^{k+1}|V|)$, given in [2].

The algorithm given in Section 5 of [18] solves a problem closely related to k'-WEC, and can be easily modified to solve k'-WEC in time $O(2.851^{k'}|\mathcal{S}||U|\cdot\log^2|U|)$. We call this algorithm $\mathsf{WECAlg}(U,\mathcal{S},w,k',p')$.

Let Π be an FGPP having parameters α_1 and α_2. We now describe algorithm $\mathsf{FGPPAlg}$ (see the pseudocode below). First, if Π is a degrading FGPP, then $\mathsf{FGPPAlg}$ solves Π by calling DegAlg. Otherwise, by using the reduction f, $\mathsf{FGPPAlg}$ transforms the input into an instance of k'-WEC. Then, $\mathsf{FGPPAlg}$ decreases the size of the resulting instance by calling the procedure $\mathsf{Decrease}$. Finally, $\mathsf{FGPPAlg}$ solves Π by calling WECAlg.

[2] This result builds on a powerful construction technique for representative families presented in [10].

Algorithm 1 $\mathsf{FGPPAlg}(G=(V,E),k,p)$

1: **if** (Π is a max-FGPP and $\frac{\alpha_1}{2}\leq\alpha_2$) or ($\Pi$ is a min-FGPP and $\frac{\alpha_1}{2}\geq\alpha_2$) **then**
2: **accept** iff $\mathsf{DegAlg}(G,k,p)$ accepts.
3: **end if**
4: $(U,\mathcal{S},w,k',p')\Leftarrow f(G,k,p)$.
5: $(U,\widehat{\mathcal{S}},w,k',p')\Leftarrow \mathsf{Decrease}(U,\mathcal{S},w,k',p')$.
6: **accept** iff $\mathsf{WECAlg}(U,\widehat{\mathcal{S}},w,k',p')$ accepts.

Theorem 2. *Algorithm* FGPPAlg *solves* Π *in time* $O(4^{k+o(k)}\Delta^k(|V|+|E|)|V|)$.

Proof. The correctness of the algorithm follows immediately from Lemmas 1 and 4, and the correctness of DegAlg and WECAlg.

By Lemmas 3 and 5, and the running times of DegAlg and WECAlg, algorithm FGPPAlg runs in time

$$\begin{aligned}
&O(4^k(\Delta-1)^k(|V|+|E|)|V| + \sum_{i=1}^{k}(4^i(\Delta-1)^i|V|(\frac{2k}{2k-i})^{k-i}2^{o(k)}\log|V|)\\
&+ 2.851^k 2.5^{k+o(k)}|V|\log^3|V|)\\
&= O(4^k\Delta^k(|V|+|E|)|V| + 2^{o(k)}|V|\log|V|[\max_{0\leq\alpha\leq1}\{4^\alpha\Delta^\alpha(\frac{2}{2-\alpha})^{1-\alpha}\}]^k)\\
&= O(4^k\Delta^k(|V|+|E|)|V| + 4^{k+o(k)}\Delta^k|V|\log|V|)\\
&= O(4^{k+o(k)}\Delta^k(|V|+|E|)|V|).
\end{aligned}$$

□

3 Solving Max $(k,n-k)$ Cut in Time $O^*(4^{p+o(p)})$

We give below an $O^*(4^{p+o(p)})$ time algorithm for Max $(k,n-k)$ Cut. In Section 3.1 we show that it suffices to consider an easier variant of Max $(k,n-k)$ Cut, that we call NC-Max $(k,n-k)$-Cut. We solve this variant in Section 3.2. Finally, our algorithm for Max $(k,n-k)$ Cut is given in Section 3.3.

3.1 Simplifying Max $(k,n-k)$ Cut

We first define an easier variant of Max $(k,n-k)$ Cut. Given a graph $G=(V,E)$ in which each node is either red or blue, and positive integers k and p, NC-Max $(k,n-k)$-Cut asks if there is a subset $X\subseteq V$ of exactly k red nodes and no blue nodes, such that at least p edges in $E(X,V\setminus X)$ have a blue endpoint.

Given an instance (G,k,p) of Max $(k,n-k)$ Cut, we perform several iterations of coloring the nodes in G; thus, if (G,k,p) is a yes-instance, we generate at least one yes-instance of NC-Max $(k,n-k)$-Cut. To determine how to color the nodes in G, we need the following definition of universal sets.

Definition 2. *Let $\mathcal{F}$ be a set of functions $f:\{1,2,\ldots,n\}\rightarrow\{0,1\}$. We say that $\mathcal{F}$ is an (n,t)-universal set if for every subset $I\subseteq\{1,2,\ldots,n\}$ of size t and a function $f':I\rightarrow\{0,1\}$, there is a function $f\in\mathcal{F}$ such that for all $i\in I$, $f(i)=f'(i)$.*

The following result asserts that small universal sets can be computed efficiently.

Lemma 6 ([16]). *There is an algorithm,* UniSetAlg*, that given a pair of integers (n,t), computes an (n,t)-universal set $\mathcal{F}$ of size $2^{t+o(t)}\log n$ in time $O(2^{t+o(t)}n\log n)$.*

We now present ColorNodes (see the pseudocode below), a procedure that given an input (G,k,p,q), where (G,k,p) is an instance of Max $(k,n-k)$ Cut and $q=k+p$, returns a set of instances of NC-Max $(k,n–k)$-Cut. Procedure ColorNodes first constructs a $(|V|,k{+}p)$-universal set $\mathcal{F}$. For each $f\in\mathcal{F}$, ColorNodes generates a colored copy V^f of V. Then, ColorNodes returns a set $\mathcal{I}$, including the resulting instances of NC-Max $(k,n-k)$-Cut.

Procedure ColorNodes$(G=(V,E),k,p,q)$

```
let V = {v_1, v_2, ..., v_|V|}.
F ⇐ UniSetAlg(|V|, q).
for all f ∈ F do
    let V^f = {v_1^f, v_2^f, ..., v_|V|^f}, where v_i^f is a copy of v_i.
    for i = 1, 2, ..., |V| do
        if f(i) = 0 then color v_i^f red. else color v_i^f blue. end if
    end for
end for
return I = {(G_f = (V_f, E), k, p) : f ∈ F}.
```

The next lemma states the correctness of procedure ColorNodes.

Lemma 7. *An instance (G,k,p) of* Max $(k,n-k)$-Cut *is a yes-instance iff* ColorNodes*$(G,k,p,k+p)$ returns a set $\mathcal{I}$ containing at least one yes-instance of* NC-Max $(k,n-k)$-Cut.

Proof. If (G,k,p) is a no-instance of Max $(k,n-k)$-Cut, then clearly, for any coloring of the nodes in V, we get a no-instance of NC-Max $(k,n–k)$-Cut. Next suppose that (G,k,p) is a yes-instance, and let X be a set of k nodes in V such that $|E(X,V\setminus X)|\geq p$. Note that there is a set Y of at most p nodes in $V\setminus X$ such that $|E(X,Y)|\geq p$. Let X' and Y' denote the indices of the nodes in X and Y, respectively. Since $\mathcal{F}$ is a $(|V|,k+p)$-universal set, there is $f\in\mathcal{F}$ such that: (1) for all $i\in X'$, $f(i)=0$, and (2) for all $i\in Y'$, $f(i)=1$. Thus, in G_f, the copies of the nodes in X are red, and the copies of the nodes in Y are blue. We get that (G_f,k,p) is a yes-instance of NC-Max $(k,n-k)$-Cut. □

Furthermore, Lemma 6 immediately implies the following result.

Lemma 8. *Procedure* ColorNodes *runs in time $O(2^{q+o(q)}|V|\log|V|)$, and returns a set $\mathcal{I}$ of size $O(2^{q+o(q)}\log|V|)$.*

3.2 A Procedure for NC-Max $(k,n-k)$-Cut

We now present SolveNCMaxCut, a procedure for solving NC-Max $(k,n–k)$-Cut (see the pseudocode below). Procedure SolveNCMaxCut orders the red nodes in V according to the number of their blue neighbors in a non-increasing manner. If there are at least k red nodes, and the number of edges between the first k red nodes and blue nodes is at least p, procedure SolveNCMaxCut accepts; otherwise, procedure SolveNCMaxCut rejects.

Clearly, the following result concerning SolveNCMaxCut is correct.

Lemma 9. *Procedure* SolveNCMaxCut *solves* NC-Max $(k,n-k)$-Cut *in time $O(|V|\log|V|+|E|)$.*

Procedure $\mathsf{SolveNCMaxCut}(G\!=\!(V,E),k,p)$

```
for all red v∈V do compute the number n_b(v) of blue neighbors of v in G. end for
let v_1,v_2,...,v_r, for some 0 ≤ r ≤ |V|, denote the red nodes in V, such that
   n_b(v_i) ≥ n_b(v_{i+1}) for all 1≤i≤r−1.
accept iff (r≥k and Σ_{i=1}^{k} n_b(v_i) ≥ p).
```

3.3 An Algorithm for Max $(k, n-k)$ Cut

Assume w.l.o.g that G has no isolated nodes. Our algorithm, MaxCutAlg, for Max $(k,n-k)$ Cut, proceeds as follows. First, if $p < \min\{k,|V|-k\}$, then MaxCutAlg accepts, and if $|V|-k < k$, then MaxCutAlg calls itself with $|V|-k$ instead of k. Then, MaxCutAlg calls ColorNodes to compute a set of instances of NC-Max $(k,n-k)$-Cut, and accepts iff SolveNCMaxCut accepts at least one of them.

Algorithm 2 $\mathsf{MaxCutAlg}(G\!=\!(V,E),k,p)$

```
1: if p < min{k,|V|−k} then accept. end if
2: if |V|−k < k then accept iff MaxCutAlg(G,|V|−k,p) accepts. end if
3: I ⇐ ColorNodes(G,k,p,k+p).
4: for all (G',k',p') ∈ I do
5:    if SolveNCMaxCut(G',k',p') accepts then accept. end if
6: end for
7: reject.
```

The next lemma implies the correctness of Step 1 in MaxCutAlg.

Lemma 10 ([2]). *In a graph $G=(V,E)$ having no isolated nodes, there is a subset $X\subseteq V$ of size k such that $|E(X,V\setminus X)| \geq \min\{k,|V|-k\}$.*

Our main result is the following.

Theorem 3. *Algorithm* MaxCutAlg *solves* Max $(k,n–k)$ Cut *in time $O(4^{p+o(p)}\cdot(|V|+|E|)\log^2|V|)$.*

Proof. Clearly, (G,k,p) is a yes-instance iff $(G,|V|-k,p)$ is a yes-instance. Thus, Lemmas 7, 9 and 10 immediately imply the correctness of MaxCutAlg.

Denote $m=\min\{k,|V|-k\}$. If $p<m$, then MaxCutAlg runs in time $O(1)$. Next suppose that $p\geq m$. Then, by Lemmas 8 and 9, MaxCutAlg runs in time $O(2^{m+p+o(m+p)}(|V|+|E|)\log^2|V|) = O(4^{p+o(p)}(|V|+|E|)\log^2|V|)$. ◻

4 Solving Positive Min-FGPPs in Time $O^*(2^{k+\frac{p}{\alpha_2}+o(k+p)})$

Let Π be a min-FGPP satisfying $\alpha_1\geq 0$ and $\alpha_2>0$. In this section we develop an $O^*(2^{k+\frac{p}{\alpha_2}+o(k+p)})$ time algorithm for Π. Using randomized separation, we show in Section 4.1 that we can focus on an easier version of Π. We solve this version in Section 4.2, using dynamic programming. Then, Section 4.3 gives our algorithm.

4.1 Simplifying the Positive Min-FGPP Π

We first define an easier variant of Π. Given a graph $G=(V,E)$ in which each node is either red or blue, and parameters $k\in\mathbb{N}$ and $p\in\mathbb{R}$, NC-Π asks if there is a subset $X\subseteq V$ of exactly k red nodes and no blue nodes, whose neighborhood outside X includes only blue nodes, such that $\mathrm{val}(X)\leq p$.

The simplification process is similar to that performed in Section 3.1. However, we now use the randomized separation procedure ColorNodes, defined in Section 3.1, with instances of Π, and consider the set $\mathcal{I}$ returned by ColorNodes as a set of instances of NC-Π. We next prove that ColorNodes is correct.

Lemma 11. *An instance (G,k,p) of Π is a yes-instance iff* ColorNodes*$(G,k,p,k+\frac{p}{\alpha_2})$ returns a set $\mathcal{I}$ containing at least one yes-instance of* NC-Π*.*

Proof. If (G,k,p) is a no-instance of Π, then clearly, for any coloring of the nodes in V, we get a no-instance of NC-Π. Next suppose that (G,k,p) is a yes-instance, and let X be a set of k nodes in V such that $\mathrm{val}(X)\leq p$. Let Y denote the neighborhood of X outside X. Note that $|Y|\leq\frac{p}{\alpha_2}$. Let X' and Y' denote the indices of the nodes in X and Y, respectively. Since $\mathcal{F}$ is a $(|V|,k+\frac{p}{\alpha_2})$-universal set, there is $f\in\mathcal{F}$ such that: (1) for all $i\in X'$, $f(i)=0$, and (2) for all $i\in Y'$, $f(i)=1$. Thus, in G_f, the copies of the nodes in X are red, and the copies of the nodes in Y are blue. We get that (G_f,k,p) is a yes-instance of NC-Π. $\square$

4.2 A Procedure for NC-Π

We now present SolveNCP, a dynamic programming-based procedure for solving NC-Π (see the pseudocode below). Procedure SolveNCP first computes the node-sets of the maximal connected red components in G. Then, procedure SolveNCP generates a matrix M, where each entry $[i,j]$ holds the minimum value $\mathrm{val}(X)$ of a subset $X\subseteq V$ in $Sol_{i,j}$, the family containing every set of exactly j nodes in V obtained by choosing a union of sets in $\{C_1,C_2\dots,C_i\}$, i.e., $Sol_{i,j}=\{(\bigcup\mathcal{C}'):\mathcal{C}'\subseteq\{C_1,C_2,\dots,C_i\},|\bigcup\mathcal{C}'|=j\}$. Procedure SolveNCP computes M by using dynamic programming, assuming an access to a non-existing entry returns ∞, and accepts iff $\mathrm{M}[t,k]\leq p$.

Procedure SolveNCP$(G=(V,E),k,p)$

```
use DFS to compute the family C = {C_1, C_2, ..., C_t}, for some 0 ≤ t ≤ |V|, of the
   node-sets of the maximal connected red components in G.
let M be a matrix containing an entry [i, j] for all 0 ≤ i ≤ t and 0 ≤ j ≤ k.
initialize M[i, 0] ⇐ 0 for all 0 ≤ i ≤ t, and M[0, j] ⇐ ∞ for all 1 ≤ j ≤ k.
for i = 1, 2, ..., t, and j = 1, 2, ..., k do
   M[i, j] ⇐ min{M[i−1, j], M[i−1, j−|C_i|] + val(C_i)}.
end for
accept iff M[t, k] ≤ p.
```

The following lemma states the correctness and running time of SolveNCP.

Lemma 12. *Procedure* SolveNCP *solves* NC-Π *in time $O(|V|k+|E|)$.*

Proof. For all $0 \leq i \leq t$ and $0 \leq j \leq k$, denote $\mathrm{val}(i,j) = \min_{X \in Sol_{i,j}}\{\mathrm{val}(X)\}$. Using a simple induction on the computation of M, we get that $\mathrm{M}[i,j] = \mathrm{val}(i,j)$. Since (G,k,p) is a yes-instance of NC-Π iff $\mathrm{val}(t,k) \leq p$, we have that SolveNCP is correct. Step 1, and the computation of $\mathrm{val}(C)$ for all $C \in \mathcal{C}$, are performed in time $O(|V|+|E|)$. Since M is computed in time $O(|V|k)$, we have that SolveNCP runs in time $O(|V|k+|E|)$. ◻

4.3 An Algorithm for Π

We now conclude PAlg, our algorithm for Π (see the pseudocode below). Algorithm PAlg calls ColorNodes to compute several instances of NC-Π, and accepts iff SolveNCP accepts at least one of them.

Algorithm 3 PAlg$(G = (V, E), k, p)$

1: $\mathcal{I} \Leftarrow$ ColorNodes$(G, k, p, k + \frac{p}{\alpha_2})$.
2: **for all** $(G', k', p') \in \mathcal{I}$ **do**
3:    **if** SolveNCP(G', k', p') accepts **then accept**. **end if**
4: **end for**
5: **reject**.

By Lemmas 8, 11 and 12, we have the following result.

Theorem 4. *Algorithm* PAlg *solves Π in time* $O(2^{k+\frac{p}{\alpha_2}+o(k+p)}(|V|+|E|)\log|V|)$.

5 Solving a Subclass of Positive Min-LGPPs Faster

Let Π be a min-FGPP satisfying $\alpha_2 \geq \frac{\alpha_1}{2} > 0$. Denote $x = \max\{\frac{p}{\alpha_2}, \min\{\frac{p}{\alpha_1}, \frac{p}{\alpha_2} + (1 - \frac{\alpha_1}{\alpha_2})k\}\}$. In this section we develop an $O^*(2^{x+o(x)})$ time algorithm for Π, that is faster than the algorithm in Section 4. Applying a divide-and-conquer step to the edges in the input graph G, Section 5.1 shows that we can focus on an easier version of Π. This version is solved in Section 5.2 by using dynamic programming. We give the algorithm in Section 5.3.

5.1 Simplifying the Non-Degrading Positive Min-FGPP Π

We first define an easier variant of Π. Suppose we are given a graph $G = (V,E)$ in which each edge is either red or blue, and parameters $k \in \mathbb{N}$ and $p \in \mathbb{R}$. For any subset $X \subseteq V$, let $\mathrm{C}(X)$ denote the family containing the node-sets of the maximal connected components in the graph $G_r = (X, E_r)$, where E_r is the set of red edges in E having both endpoints in X. Also, let $\mathrm{val}^*(X) = \sum_{C \in \mathrm{C}(X)} \mathrm{val}(C)$. The variant EC-$\Pi$ asks if there is a subset $X \subseteq V$ of exactly k nodes, such that all the edges in $E(X, V \setminus X)$ are blue, and $\mathrm{val}^*(X) \leq p$.

We now present a procedure, called ColorEdges (see the pseudocode below), whose input is an instance (G,k,p) of Π. Procedure ColorEdges uses a universal set to perform several iterations coloring the edges in G, and then returns the resulting set of instances of EC-Π.

The following lemma states the correctness of ColorEdges.

Procedure ColorEdges$(G=(V,E),k,p)$

let $E=\{e_1,e_2,\ldots,e_{|E|}\}$.
$\mathcal{F} \Leftarrow$ UniSetAlg$(|E|,x)$.
for all $f \in \mathcal{F}$ **do**
  let $E^f=\{e^f_1,e^f_2,\ldots,e^f_{|E|}\}$, where e^f_i is a copy of e_i.
  for $i=1,2,\ldots,|E|$ **do**
    if $f(i)=0$ **then** color e^f_i red. **else** color e^f_i blue. **end if**
  end for
end for
return $\mathcal{I}=\{(G_f=(V,E_f),k,p) : f\in\mathcal{F}\}$.

Lemma 13. *An instance* (G,k,p) *of* Π *is a yes-instance iff* ColorEdges(G,k,p) *returns a set* $\mathcal{I}$ *containing at least one yes-instance of* EC-Π.

Proof. Since $\alpha_2 \geq \frac{\alpha_1}{2}$, $\mathrm{val}^*(X) \geq \mathrm{val}(X)$ for any set $X \subseteq V$ and coloring of edges in E. Thus, if (G,k,p) is a no-instance of Π, then clearly, for any coloring of edges in E, we get a no-instance of EC-Π. Next suppose that (G,k,p) is a yes-instance, and let X be a set of k nodes in V such that $\mathrm{val}(X)\leq p$. Let $\widetilde{E}_r=E(X)$, and $E_b=E(X,V\setminus X)$. Also, choose a minimum-size subset $E_r\subseteq\widetilde{E}_r$ such that the graphs $G'_r=(X,\widetilde{E}_r)$ and $G_r=(X,E_r)$ contain the same set of maximal connected components. Let E'_r and E'_b denote the indices of the edges in E_r and E_b, respectively. Note that $|E'_r|+|E'_b|\leq x$. Since $\mathcal{F}$ is an $(|E|,x)$-universal set, there is $f\in\mathcal{F}$ such that: (1) for all $i\in E'_r$, $f(i)=0$, and (2) for all $i\in E'_b$, $f(i)=1$. Thus, in G_f, the copies of the edges in E_r are red, and the copies of the edges in E_b are blue. Then, $\mathrm{val}^*(X)=\mathrm{val}(X)$. We get that (G_f,k,p) is a yes-instance of EC-Π. ◻

Furthermore, Lemma 6 immediately implies the following result.

Lemma 14. *Procedure* ColorEdges *runs in time* $O(2^{x+o(x)}|E|\log|E|)$, *and returns a set* $\mathcal{I}$ *of size* $O(2^{x+o(x)}\log|E|)$.

5.2 A Procedure for EC-Π

By modifying the procedure given in Section 4.2, we get a procedure, called SolveECP, satisfying the following result (see Appendix B).

Lemma 15. *Procedure* SolveECP *solves* EC-Π *in time* $O(|V|k+|E|)$.

5.3 A Faster Algorithm for Π

Our faster algorithm for Π, FastPAlg, calls ColorEdges to compute several instances of EC-Π, and accepts iff SolveECP accepts at least one of them (see the pseudocode below).

By Lemmas 13, 14 and 15, we have the following result.

Theorem 5. *Algorithm* FastPAlg *solves* Π *in time* $O(2^{x+o(x)}(|V|k+|E|)\log|E|)$.

Since Min k-Vertex Cover satisfies $\alpha_1=\alpha_2=1$, we have the following result.

Corollary 1. *Algorithm* FastPAlg *solves* Min k-Vertex Cover *in time* $O(2^{p+o(p)}(|V|k+|E|)\log|E|)$.

Algorithm 4 FastPAlg($G = (V, E), k, p$)

1: $\mathcal{I} \Leftarrow$ ColorEdges(G, k, p).
2: **for all** $(G', k', p') \in \mathcal{I}$ **do**
3: **if** SolveECP(G', k', p') accepts **then accept**. **end if**
4: **end for**
5: **reject**.

References

1. Berkhin, P.: A survey of clustering data mining techniques. Grouping Multidimensional Data Recent Advances in Clustering, Eds. J. Kogan and C. Nicholas and M. Teboulle pp. 25–71 (2006)
2. Bonnet, E., Escoffier, B., Paschos, V.T., Tourniaire, E.: Multi-parameter complexity analysis for constrained size graph problems: using greediness for parameterization. In: IPEC. pp. 66–77 (2013)
3. Bourgeois, N., Giannakos, A., Lucarelli, G., Milis, I., Paschos, V.T.: Exact and approximation algorithms for densest k-subgraph. In: IPEC. pp. 66–77 (2013)
4. Cai, L.: Parameterized complexity of cardinality constrained optimization problemss. Comput. J. 51(1), 102–121 (2008)
5. Cai, L., Chan, S.M., Chan, S.O.: Random separation: A new method for solving fixed-cardinality optimization problems. In: IPEC. pp. 239–250 (2006)
6. Cygan, M., Lokshtanov, D., Pilipczuk, M., Pilipczuk, M., Saurabh, S.: Minimum bisection is fixed parameter tractable. In: STOC (2014, to appear)
7. Donavalli, A., Rege, M., Liu, X., Jafari-Khouzani, K.: Low-rank matrix factorization and co-clustering algorithms for analyzing large data sets. In: ICDEM. pp. 272–279 (2010)
8. Downey, R.G., Estivill-Castro, V., Fellows, M.R., Prieto, E., Rosamond, F.A.: Cutting up is hard to do: the parameterized complexity of k-cut and related problems. Electr. Notes Theor. Comput. Sci. 78, 209–222 (2003)
9. Downey, R.G., Fellows, M.R.: Fixed-parameter tractability and completeness II: on completeness for W[1]. Theor. Comput. Sci. 141(1&2), 109–131 (1995)
10. Fomin, F.V., Lokshtanov, D., Saurabh, S.: Efficient computation of representative sets with applications in parameterized and exact agorithms. In: SODA. pp. 142–151 (2014)
11. Guo, J., Niedermeier, R., Wernicke, S.: Parameterized complexity of vertex cover variants. Theory Comput. Syst. 41(3), 501–520 (2007)
12. Kahng, A.B., Lienig, J., Markov, I.L., Hu, J.: VLSI Physical Design - From Graph Partitioning to Timing Closure. Springer (2011)
13. Kloks, T.: Treewidth, computations and approximations. LNCS 842, Springer (1994)
14. Kneis, J., Langer, A., Rossmanith, P.: Improved upper bounds for partial vertex cover. In: WG. pp. 240–251 (2008)
15. Komusiewicz, C., Sorge, M.: Finding dense subgraphs of sparse graphs. In: IPEC. pp. 242–251 (2012)
16. Naor, M., Schulman, L.J., Srinivasan, A.: Splitters and near-optimal derandomization. In: FOCS. pp. 182–191 (1995)
17. Shachnai, H., Zehavi, M.: Faster computation of representative families for uniform matroids with applications. CoRR abs/1402.3547 (2014)
18. Zehavi, M.: Deterministic parameterized algorithms for matching and packing problems. CoRR abs/1311.0484 (2013)

A An Illustration of the Reduction f

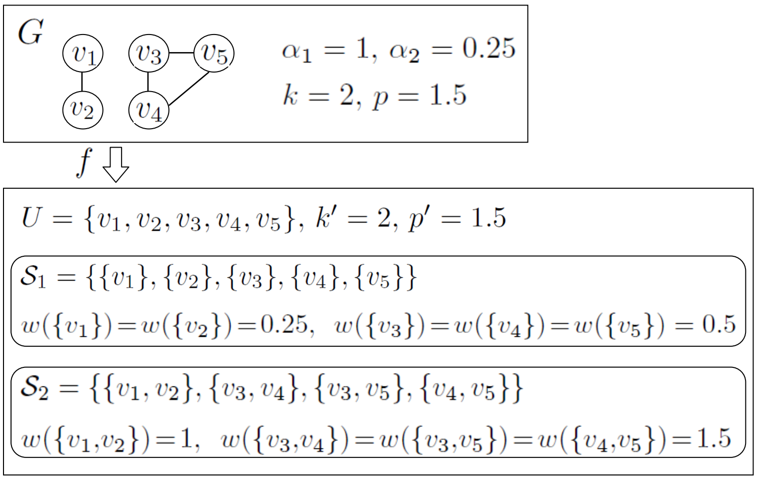

Fig. 1. An illustration of the reduction f, given in Section 2.1.

B A Procedure for EC-$\boldsymbol{\Pi}$ (Cont.)

We now present the details of procedure SolveECP (see the pseudocode below). Procedure SolveECP first computes the node-sets of the maximal connected components in the graph obtained by removing all the blue edges from G. Then, procedure SolveECP generates a matrix M, where each entry $[i, j]$ holds the minimum value $\mathrm{val}^*(X)$ of a subset $X \subseteq V$ in $Sol_{i,j}$, the family containing every set of exactly j nodes in V obtained by choosing a union of sets in $\{C_1, C_2 \ldots, C_i\}$, i.e., $Sol_{i,j} = \{(\bigcup \mathcal{C}') : \mathcal{C}' \subseteq \{C_1, C_2, \ldots, C_i\}, |\bigcup \mathcal{C}'| = j\}$. Procedure SolveNCP computes M by using dynamic programming, assuming an access to a non-existing entry returns ∞, and accepts iff $\mathrm{M}[t, k] \leq p$.

Procedure SolveECP$(G = (V, E), k, p)$

use DFS to compute the family $\mathcal{C} = \{C_1, C_2, \ldots, C_t\}$, for some $0 \leq t \leq |V|$, of the node-sets of the maximal connected components in the graph obtained by removing all the blue edges from G.
let M be a matrix containing an entry $[i, j]$ for all $0 \leq i \leq t$ and $0 \leq j \leq k$.
initialize $\mathrm{M}[i, 0] \Leftarrow 0$ for all $0 \leq i \leq t$, and $\mathrm{M}[0, j] \Leftarrow \infty$ for all $1 \leq j \leq k$.
for $i = 1, 2, \ldots, t$, and $j = 1, 2, \ldots, k$ **do**
  $\mathrm{M}[i, j] \Leftarrow \min\{\mathrm{M}[i-1, j], \mathrm{M}[i-1, j-|C_i|] + \mathrm{val}^*(C_i)\}$.
end for
accept iff $\mathrm{M}[t, k] \leq p$.

We next prove the correctness of Lemma 15.

Proof. For all $0 \leq i \leq t$ and $0 \leq j \leq k$, denote $\mathrm{val}(i,j) = \min_{X \in Sol_{i,j}} \{\mathrm{val}^*(X)\}$. Using a simple induction on the computation of M, we get that $\mathrm{M}[i,j] = \mathrm{val}(i,j)$.

Since (G,k,p) is a yes-instance of EC-Π iff $\text{val}(t,k) \leq p$, we have that SolveECP is correct. Step 1, and the computation of $\text{val}^*(C)$ for all $C \in \mathcal{C}$, are performed in time $O(|V|+|E|)$. Since M is computed in time $O(|V|k)$, we have that SolveECP runs in time $O(|V|k+|E|)$. $\square$

C Some Proofs

Proof of lemma 4: First, assume that $(U,\mathcal{S},w,k',p')$ is a yes-instance. Let $\mathcal{S}'$ be a subfamily of disjoint sets from $\mathcal{S}$, such that $|\bigcup \mathcal{S}'| = k'$, $\sum_{S \in \mathcal{S}'} w(S) \geq p'$, and there is no subfamily $\mathcal{S}''$ satisfying these conditions, and $|\mathcal{S}' \cap \widehat{\mathcal{S}}| < |\mathcal{S}'' \cap \widehat{\mathcal{S}}|$. Suppose, by way of contradiction, that there is a set $S \in (\mathcal{S}_i \cap \mathcal{S}') \setminus \widehat{\mathcal{S}}$, for some $1 \leq i \leq k'$. By Theorem 1, there is a set $\widehat{S} \in \widehat{\mathcal{S}}_i$ such that $w(\widehat{S}) \geq w(S)$, and $\widehat{S} \cap S' = \emptyset$, for all $S' \in \mathcal{S}' \setminus \{S\}$. Thus, $\mathcal{S}'' = (\mathcal{S}' \setminus \{S\}) \cup \{\widehat{S}\}$ is a solution to $(U,\mathcal{S},w,k',p')$. Since $|\mathcal{S}' \cap \widehat{\mathcal{S}}| < |\mathcal{S}'' \cap \widehat{\mathcal{S}}|$, this is a contradiction.

Now, assume that $(U,\widehat{\mathcal{S}},w,k',p')$ is a yes-instance. Since $\widehat{\mathcal{S}} \subseteq \mathcal{S}$, we immediately get that $(U,\mathcal{S},w,k',p')$ is also a yes-instance. $\square$