Laboratoire d'Analyses et Modélisation de Systèmes pour l'Aide à la Décision
UMR 7243

CAHIER DU LAMSADE

330

Décembre 2012

Using greediness for parameterization: the case of max and min (k, n - k)-cut

E. Bonnet, B. Escoffier, V.Th. Paschos, E. Tourniaire

Using greediness for parameterization: the case of max and min $(k, n-k)$-cut*

Edouard Bonnet, Bruno Escoffier, Vangelis Th. Paschos[(a)], Emeric Tourniaire

PSL Research University, Université Paris-Dauphine, LAMSADE
CNRS, UMR 7243, France
edouard.bonnet@dauphine.fr,{escoffier,paschos}lamsade.dauphine.fr
emeric.tourniaire@lamsade.dauphine.fr

December 11, 2012

Abstract

MAX $(k, n-k)$-CUT (resp., MIN $(k, n-k)$-CUT) is a constrained version of MAX-CUT (resp., MIN-CUT) where one has to find a bipartition of the vertex set into two subsets with respectively k and $n-k$ vertices (n being the total number of vertices of the input graph) which maximizes (resp., minimizes) the number of edges going from one subset to the other. In this paper, we investigate the parameterized complexity of these two graph problems by considering several parameters, such as the value p of the solution, k, the size τ of a minimum vertex cover and the treewidth tw of the input graph. We also give approximation schemata in FPT time for parameterizations which turn out to be W[1]-hard.

1 Introduction

Given a graph $G = (V, E)$ and two integers p and k, the parameterized version of the MAX $(k, n-k)$-CUT problem consists in deciding whether there exists, or not, a subset $V' \subseteq V$ such that $|V'| = k$ and $|E(V', V \setminus V')| \geqslant p$, where $E(A, B) = \{xy \in E : x \in A \wedge y \in B\}$, while the MIN $(k, n-k)$-CUT problem consists in deciding whether there exists, or not, a subset $V' \subseteq V$ such that $|V'| = k$ and $|E(V', V \setminus V')| \leqslant p$.

Both problems belong to a large set of problems, sometimes called fixed cardinality problems, where one wishes to optimize (maximize or minimize) the number of edges incident to exactly k vertices and satisfying some property. MAX k-VERTEX COVER, k-DENSEST SUBGRAPH, k-LIGHTEST SUBGRAPH, MAX $(k, n-k)$-CUT and MIN $(k, n-k)$-CUT are the most known fixed cardinality problems.

When dealing with MAX $(k, n-k)$-CUT and MIN $(k, n-k)$-CUT, two natural parameters come immediately in mind, k and the value p of the solution $E(V', V \setminus V')$ (frequently called standard parameter). More about parameterized complexity can be found in [7]. The problems handled in this paper have mainly been studied in [3, 6], from a parameterized point of view, and have been proved W[1]-hard when parameterized by k, while complexity of standard parameterization remained open.

On the other hand, approximation of MIN $(k, n-k)$-CUT has been studied in [8] where it is proved that, if $k = O(\log n)$, then the problem admits a randomized polynomial time approximation schema, while, if $k = \Omega(\log n)$, then it admits an approximation ratio $(1 + \frac{\varepsilon k}{\log n})$, for any $\varepsilon > 0$. Approximation of MAX $(k, n-k)$-CUT has been studied in several papers and a ratio $1/2$ is achieved in [1] (slightly improved with a randomized algorithm in [9]), for all k.

Here, we first consider the parameterized complexity of MAX and MIN $(k, n-k)$-CUT with respect to the standard parameter p and prove that the former problem is fixed parameter tractable (FPT),

*Research supported by the French Agency for Research under the program TODO, ANR-09-EMER-010
[(a)]Institut Universitaire de France

while, unfortunately, the status of the latter one remains still unclear (Section 2). In order to handle MAX $(k, n-k)$-CUT we first show that when $p \leqslant k$ or $p \leqslant \Delta$, where Δ denotes the maximum degree of G, the problem is polynomial. So, the only "non-trivial" case occurs when $p > k$ and $p > \Delta$. We then design an FPT algorithm parameterized by $k\Delta$ which allows us to conclude that MAX $(k, n-k)$-CUT parameterized by p is FPT. Then, a refinement of this algorithm allows to show that MIN $(k, n-k)$-CUT parameterized by $k\Delta$ is also FPT. Unfortunately, this shows inclusion in FPT of MIN $(k, n-k)$-CUT only for some particular cases.

To achieve these FPT algorithms, which are the main contributions of this article, we use the following original idea, which might also be useful for other problems. It consists of performing a branching with respect to a vertex chosen upon some greedy criterion. For MAX $(k, n-k)$-CUT this criterion is to consider some vertex v that maximizes the number of edges added to the cut under construction. Without branching, the greedy algorithm is not optimal. However, the principle is that at each step either the greedily chosen vertex v is a good choice (it is in an optimal solution), or some of its neighbors (or a vertex at bounded distance from v) is a good choice (it is an optimal solution). This induces a branching rule on neighbors of v which leads to a branching tree whose size is, in our case, bounded by a function of k and Δ.

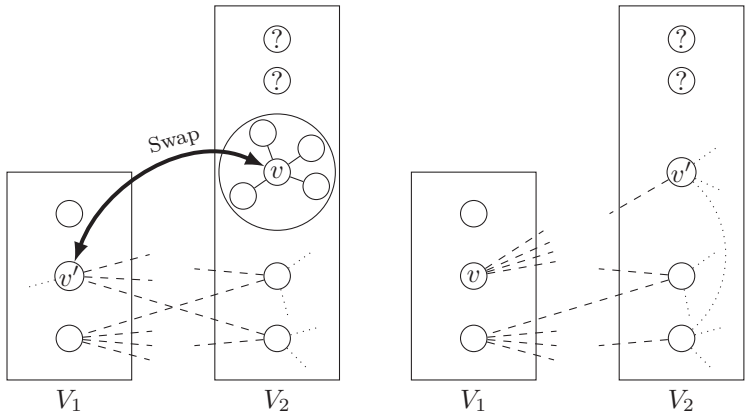
In Section 3, we mainly revisit the parameterization by k but we handle it from an approximation point of view. Given a problem Π parameterized by parameter ℓ and an instance I of Π, a parameterized approximation algorithm for Π is an algorithm running in time $f(\ell)|I|^{O(1)}$ that either finds an approximate solution of size $g(\ell)$ as close as possible to ℓ, or reports that there is no solution of size ℓ. We prove that, although W[1]-hard for the exact computation, MAX $(k, n-k)$-CUT has a parameterized approximation schema with respect to k and MIN $(k, n-k)$-CUT a randomized parameterized approximation schema. These results exhibit two problems which are hard with respect to a given parameter but which become easier when we relax exact computation requirements and seek only (good) approximations. To our knowledge, the only other problem having similar behaviour is another fixed cardinality problem, the MAX k-VERTEX COVER problem [13]. Note that the existence of problems having this behaviour but with respect to the standard parameter is an open (presumably very difficult to answer) question in [13].

In Section 4, we first handle parameterization of both problems by the treewidth tw of the input graph and show, using a standard dynamic programming technique, that they admit an $O^*(2^{\mathrm{tw}})$-time FPT algorithm, when the $O^*(\cdot)$ notation ignores polynomial factors. Indeed, the result is stronger as we prove that a whole class of problems, including fixed cardinality problems but also, for instance MIN BISECTION problem can be solved by the same algorithm. Let us note that the interest of this result, except its structural aspect (many problems for the price of a single algorithm), lies also in the fact that MAX and MIN k-VERTEX COVER does not fit Courcelle's Theorem [5]. Indeed, MAX and MIN BISECTION are not expressible in MSO since the equality of the cardinality of two sets is not MSO-definable. In fact, if one could express that two sets have the same cardinal in MSO, one would be able to express in MSO the fact that a word has the same number of a's and b's, on a two-letter alphabet, which would make that the set $E = \{w : |w|_a = |w|_b\}$ is MSO-definable. But we know that, on words, MSO-definability is equivalent to recognizability; we also know by the standard pumping lemma (see, for instance, [11]) that E is not recognizable [12], a contradiction. Henceforth, MAX and MIN k-VERTEX COVER are no more expressible in MSO; consequently, the fact that those two problems, parameterized by tw are FPT cannot be obtained by Courcelle's Theorem. Furthermore, even several known extended variants of MSO which capture more problems [14], does not seem to be able to express the equality of two sets either.

2 Standard parameterization

2.1 Max $(\mathbf{k}, \mathbf{n-k})$-cut

In the sequel, we denote by $N(v)$ the set of neighbors of v in $G = (V, E)$, namely $\{w \in V: \{v, w\} \in E\}$ and define $N[v] = N(v) \cup \{v\}$. We also use the standard notation $G[U]$ for any $U \subseteq V$ to denote the subgraph induced by the vertices of U. In this section, we show that MAX $(k, n-k)$-CUT parameterized by the standard parameter, i.e., by the value p of the solution, is FPT. Using an idea of bounding above the value of an optimal solution by a swapping process (see



(a) Vertices $v \in V_2$ and $v' \in V_1$ (that has at least one neighbor in V_1) will be swapped.

(b) With the swapping the cut size increases.

Figure 1: Illustration of a swapping

Figure 1), we show that the non trivial case satisfies $p > k$. We also show that $p > \Delta$ holds for non trivial instances and get the situation depicted by Figure 2. The rest of the proof (see Theorem 3) shows that MAX $(k, n-k)$-CUT parameterized by $k\Delta$ is FPT, by designing a particular branching algorithm. This branching algorithm is based on the following intuitive idea. Consider a vertex v of maximum degree in the graph. If an optimal solution $E(V', V \setminus V')$ is such that no vertex of $N(v)$ is in V', then it is always interesting to take v in V' (this provides Δ edges to the cut, which is the best we can do). This leads to a branching rule with $\Delta + 1$ branches, where in each branch we take in V' one vertex from $N[v]$.

Lemma 1. *In a graph with minimum degree r, the optimal value* opt *of a* MAX $(k, n-k)$-CUT *satisfies* opt $\geqslant \min\{n-k, rk\}$.

Proof. We divide arbitrarily the vertices of a graph $G = (V, E)$ into two subsets V_1 and V_2 of size k and $n-k$, respectively. Then, for every vertex $v \in V_2$, we check if v has a neighbor in V_1. If not, we try to swap v and a vertex $v' \in V_1$ which has strictly less than r neighbors in V_2 (see Figure 1). If there is no such vertex, then every vertex in V_1 has at least r neighbors in V_2, so determining a cut of value at least rk. When swapping is possible, as the minimum degree is r and the neighborhood of v is entirely contained in V_2, moving v from V_2 to V_1 will increase the value of the cut by at least r. On the other hand, moving v' from V_1 to V_2 will reduce the value of the cut by at most $r-1$. In this way, the value of the cut increases by at least 1.

Finally, either the process has reached a cut of value rk (if no swap is possible), or every vertex in V_2 has increased the value of the cut by at least 1 (either immediately, or after a swapping process), which results in a cut of value at least $n-k$, and the proof of the lemma is completed. □

Corollary 2. *In a graph with no isolated vertices, the optimal value for* MAX $(k, n-k)$-CUT *is at least* $\min\{n-k, k\}$.

Theorem 3. *The* MAX $(k, n-k)$-CUT *problem parameterized by the standard parameter p is FPT.*

Proof. Considering the symmetry between V' and $V \setminus V'$, one can assume that $k \leqslant \frac{n}{2} \leqslant n-k$. If G contains i isolated vertices, one can run the algorithm we are describing with k' going from $\min\{0, k-i\}$ to k, in G' that is G without the i isolated vertices, with the same value for p, and output YES, if one of the calls outputs YES. Thus, we consider that G contains no isolated vertices.

In addition, if $p > \frac{n}{2}$, then we may exhaustively compute all the $(k, n-k)$-cuts, by computing all the subsets of V on k vertices and the cut any of them induces, in time $O^*(2^n)$. Consequently, in this case, the time needed for the work is FPT, $O^*(2^{2p})$. So, in the sequel, we will assume $p \leqslant \frac{n}{2}$. Furthermore, by Corollary 2 and the discussion above, the overall assumption for the sequel (in

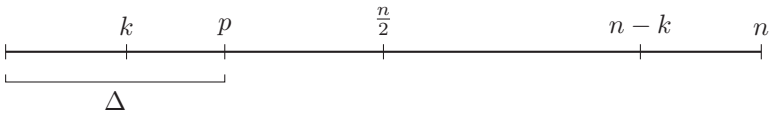


Figure 2: Location of parameter p, relatively to k and Δ.

the proof of the theorem) is $k \leqslant p \leqslant \frac{n}{2} \leqslant n-k$, since in a graph with no isolated vertices, the minimum degree is at least 1.

Denote by Δ the maximum degree in the input graph G, and let v_0 be a vertex with maximum degree. If $p \leqslant \Delta$, putting v_0 in V', setting $d = \min\{n-k, \Delta\}$ of its neighbors outside V' and completing arbitrarily (if necessary) the subset $V \setminus V'$, yields a cut of size k and of value at least d. Since $p \leqslant n-k$, we get $p \leqslant d$, thus this cut is already a solution. So, together with $k \leqslant p \leqslant \frac{n}{2} \leqslant n-k$, we can, in addition, assume $\Delta < p$.

We are ready now to specify the following branching algorithm. We consider a set T which is initially empty, and which will intuitively correspond to V' at the end. At each branching step the algorithm puts one more vertex in T. Hence, the depth of the tree is k.

To do the branching, we consider a vertex v in $V \setminus T$ which maximizes the quantity:

$$\delta_T(v) \quad = \quad |\{u \in N(v) \cap (V \setminus T)\}| - |\{u \in N(v) \cap T\}|$$

This is a greedy criterion: $\delta_T(v)$ represents the increasing of the value of the cut by taking v in T. For instance, initially $T = \emptyset$ and one can choose the vertex of maximal degree v_0.

Then, in the branching tree, the node has $|N[v] \cap (V \setminus T)|$ children: in each child we take one vertex from $N[v] \cap (V \setminus T)$ and put it in T. We stop branching in nodes where $|T| = k$ and output YES if one of the induced $(k, n-k)$-cuts has value at least p. Note that in branching steps $|N[v]|$ is bounded by $\Delta + 1$, and so is the number of sons of a node (i.e., the arity of the branching tree). Since the depth of the tree is k, the algorithm runs in $O^*(\Delta^k) = O^*(p^p)$.

Let us now show that our algorithm is sound. To prove optimality, we use a classical hybridation technique between some optimal solution and our solution. Consider an optimal solution V'_{opt} (in the sequel, we assume that a solution is represented by the subset V' of size k) different from the solution computed by the algorithm. A node w of the branching tree has two characteristics: the set of taken vertices $T(w)$ (or T if no ambiguity occurs) and the vertex chosen by the greedy criterion $v(w)$ (or simply v). We say that a node w of the branching tree is conform to the optimal solution V'_{opt} if $T(w) \subseteq V'_{\text{opt}}$. A node w deviates from the optimal solution V'_{opt} if none of its sons is conform to V'_{opt}.

We start from the root of our branching tree and we go to a conform son of our current node while it is possible. At some point we stop by reaching a node w which deviates from V'_{opt}. We set $T = T(w)$ and $v = v(w)$. Intuitively, T corresponds to the shared choices between the optimal solution and our algorithm made along the branch from the root to the node w. Given a vertex $z \in V'_{\text{opt}} \setminus T$, we consider the cut induced by the set $V'_{\text{opt}} \cup \{v\} \setminus \{z\}$ obtained from V'_{opt} by adding v and removing z. We show that this cut is also optimal. When removing z and adding v to V'_{opt}, we remove $\delta_{V'_{\text{opt}}}(z)$ edges and add $\delta_{V'_{\text{opt}}}(v)$ edges to the cut, so we only need to show that $\delta_{V'_{\text{opt}}}(v) \geqslant \delta_{V'_{\text{opt}}}(z)$ (which is then an equality by optimality of V'_{opt}). First note that $\delta_T(z) \leqslant \delta_T(v)$, by the choice of v. Moreover, since $T \subseteq V'_{\text{opt}}$, $\delta_{V'_{\text{opt}}}(z) \leqslant \delta_T(z)$ (more generally the quantity δ_T never increases since vertices are never removed from T). Furthermore, since the optimal solution does not take any vertex in $N(v) \cap (V \setminus T)$, it holds that $\delta_T(v) = \delta_{V'_{\text{opt}}}(v)$. Therefore, $\delta_{V'_{\text{opt}}}(z) \leqslant \delta_{V'_{\text{opt}}}(v)$, and the cut induced by $V'_{\text{opt}} \cup \{v\} \setminus \{z\}$ is optimal. Thus, by repeating this process at most k times, we can conclude that the solution computed by the algorithm has the same value as the optimal solution considered. □

2.2 Min $(k, n-k)$-cut

We give in this section a result analogous to that of Theorem 3 for MIN $(k, n-k)$-CUT. We devise a more "subtle" FPT algorithm in parameter $k\Delta$, with some refinements of the proof of Theorem 3.

Unfortunately, in the minimization case of $(k, n-k)$-cut, it enables us to conclude about the parameterized complexity in p, only in some particular cases.

Before presenting the algorithm, let us remark that the same branching algorithm as in the case of MAX $(k, n-k)$-CUT does not work anymore, since choosing to add a vertex in the solution V' could seem really bad locally, but could turn out to be an optimal choice if all its neighbors are taken in the solution later. In other words, it does not seem possible to design a proper notion of "contribution" of one vertex such as $\delta_T(v)$ which only increases from the moment where it is added in the solution until the end. In order to get through this difficulty, the algorithm we present has to be more involved. In particular, first it does not only assign vertices to V', but it also fixes some vertices to $V \setminus V'$ (definitely). To this purpose, we will maintain a partition of the vertices of V in three sets T (taken vertices), R (rejected vertices) and U (unmarked vertices). More importantly, instead of exploring only the neighborhood of a vertex v when branching, it explores the set of vertices at distance at most k from v.

Theorem 4. MIN $(k, n-k)$-CUT *parameterized by* $k\Delta$ *is FPT.*

Proof. Let $(G = (V, E), p, k)$ be a general instance of the MIN $(k, n-k)$-CUT problem, and Δ be the maximum degree of G. Let us first introduce the following additional notations. The set $N^k[v]$ denotes the set of vertices which are at distance at most k from v in the graph G ($N^1[v] = N[v]$); it is called k-neighborhood of v. In particular, the size of $N^k[v]$ for a given vertex v is bounded by Δ^{k+1}, thus respecting FPT-time (with respect to $k\Delta$).

Besides, one can consider an optimal solution as the union of its (maximal) connected components $V'_1, V'_2, \dots V'_h$ (with $V' = \bigcup_{i=1}^{h} V'_i$). Indeed, the contribution of a connected component V'_i is not varying as long as the solution is built and is definitively equal to $\delta(V'_i)$, where δ counts the number of outgoing edges of a subset of vertices. And the value opt associated to an optimal solution induced by V' is equal to $\Sigma_{i=1}^{h}\delta(V'_i)$. This idea seems to be good for decomposing an optimal solution, but if one also uses it for computing a solution (by adding at each step in the solution not a vertex but a whole connected component), one will be in trouble to prove optimality, since the hybridation has to preserve the size k of the solution.

The following Algorithm `MINCUT`(G,T,R,k), deals with the technical problems presented above:

> *Algorithm* 5*:* set $T = \emptyset$, $R = \emptyset$;
>
> - if $k > 0$ then
>   - for each vertex v in U and for each i from 1 to k, compute the minimum $\delta_{i,k}(v) = \min_S\{\delta(S) - 2|E(S,T)|\}$ where S is any subset of size i of $N^k[v] \cap U$ and determine k vertices $v_1, v_2, \dots, v_k$ respectively minimizing $\delta_{1,k}, \delta_{2,k}, \dots, \delta_{k,k}$; let $S_1, S_2, \dots, S_k$ be the corresponding minimizing subsets;
>   - for each i from 1 to k
>   - for each vertex v of $N^k[v_i] \cap U$ run `MINCUT`($G, T \cup \{v\}$,R,$k-1$);
>   - run `MINCUT`($G, T \cup S_i$,$R \cup (N^k[v_i] \setminus S_i)$,$k - |S_i|$);
> - else ($k = 0$), output $(T, V \setminus T)$.

Obviously, we return the minimum value of the $(k, n-k)$-cut among the solutions at the leaves of the branching tree, and we compare it to p.

Let us now analyze the complexity of the above algorithm. It is divided into two parts: the first part is computation of $v_1, v_2, \dots, v_k$ and the second part is the branching. Since $|N^k[v]| \leqslant \Delta^{k+1}$, computing $\delta_{i,k}(v)$ and determining $v_1, v_2, \dots, v_k$ takes time $O^*((\Delta^{k+1})^k)$. Now, the branching tree has arity at most $k(\Delta^k + 1)$ and depth at most k. So, it has at most $k^k(\Delta^k + 1)^k$ nodes. Then, computing the value of a cut can be done in polynomial time, and our algorithm works in FPT-time $O^*(k^k\Delta^{k^2})$.

We prove here that the algorithm computes an optimal solution by using hybridation of the optimal solution and of its solution. Let V'_{opt} be an optimal solution. First, we decompose V'_{opt} in its connected components $V'_{1,\text{opt}}, V'_{2,\text{opt}}, \dots, V'_{h,\text{opt}}$ (as mentioned above). Each node of the branching tree corresponds to some configuration (T, R, U). For instance, the root corresponds to

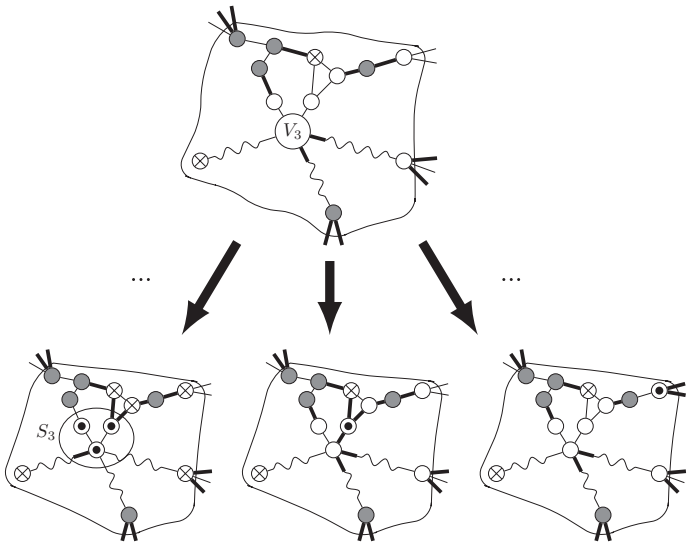


Figure 3: The k-neighborhood of v_3 in a fragment of the branching tree. Filled vertices correspond to taken vertices, crossed vertices correspond to rejected vertices, plain vertices correspond to unmarked vertices, and spotted vertices correspond to the vertices which has just been added to the taken vertices. Bold edges are the edges of the current cut. Assume the father node deviates from the optimal solution. Then, the optimal solution takes no unmarked vertices in the k-neighborhood of v_3. Thus, we can substitute S_3 which is the best local choice for adding 3 vertices, to the 3 vertices of the optimal solution which completes one connected component (see Figure 4).

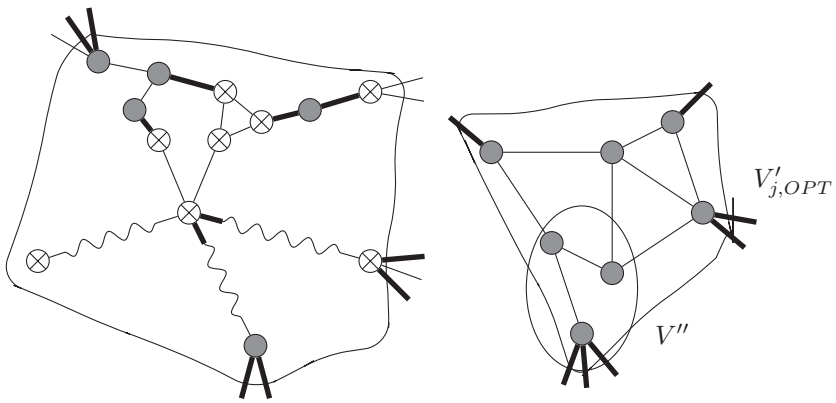


Figure 4: Representation of the optimal solution. Filled vertices are in V'_{opt} and the crossed vertices are in the other subset; $V'' = V'_{j,\text{opt}} \setminus V'_{\text{shared}}$; where $V'_{j,\text{opt}}$ is a connected component of V'_{opt} and V'_{shared} denotes the vertices both taken by the optimal solution and by `MINCUT`. As the node of the branching tree of Figure 3 is supposed to deviate from V'_{opt}, we can notice that no unmarked vertex of the extended neighborhood of v_3 is taken in V'_{opt}.

the configuration $(\emptyset, \emptyset, V)$ and any leaf corresponds to a configuration with $|T| = k$. We say that a node of the branching tree is conform to the optimal solution V'_{opt}, if its configuration (T, R, U) satisfies $T \subseteq V'_{\text{opt}}$ and $R \cap V'_{\text{opt}} = \emptyset$. Also, we say that a node of the branching tree deviates from the optimal solution when none of its children is conform to the optimal solution.

From the root, we follow any branch of the branching tree until we reach a node which deviates from V'_{opt}, and we denote by $V'_{\text{shared}} \subset V'_{\text{opt}}$ the set of vertices in the optimal solution V'_{opt} which has been taken in our solution along the followed branch. In other words, V'_{shared} corresponds to shared choices, up to this point, between our branching and the optimal solution (note that V'_{shared} and the crossway node are not necessarily unique). At this node, our algorithm computes $v_1, v_2, \ldots, v_k$ and we know for sure that, for all i, $N^k[v_i] \cap (V'_{\text{opt}} \setminus V'_{\text{shared}}) = \emptyset$. Take a whole connected component $V'_{j,\text{opt}}$ not entirely contained in V'_{shared} and let $c_j = |V'_{j,\text{opt}} \setminus V'_{\text{shared}}|$.

Let us now explain why component S_3 of Figure 3 (role played here by S) could substitute V'' of Figure 4 (role played here by $V'_{j,\text{opt}} \setminus V'_{\text{shared}}$).

By construction, in $N^k[v_{c_j}] \setminus V'_{\text{shared}}$, and by the previous remark in $N^k[v_{c_j}] \setminus V'_{\text{opt}}$, there exists a subset S with c_j elements which satisfies $\delta(S \cup V'_{\text{shared}}) \leqslant \delta(V'_{j,\text{opt}} \cup V'_{\text{shared}})$. Now, let us consider the set of k vertices $W = (V'_{\text{opt}} \setminus (V'_{j,\text{opt}} \setminus V'_{\text{shared}})) \cup S$. In the sequel, we set $V'' = V'_{j,\text{opt}} \setminus V'_{\text{shared}}$, $X = V'_{\text{shared}} \cup V'_{j,\text{opt}} = V'' \cup V'_{\text{shared}}$ and we repeatedly use that $\delta(A \cup B) = \delta(A) + \delta(B) - 2|E(A, B)|$. It holds that:

$$\delta(W) = \delta\left(S \cup V'_{\text{shared}}\right) + \delta\left(W \setminus (S \cup V'_{\text{shared}})\right) - 2\left|E\left(S \cup V'_{\text{shared}}, W \setminus (S \cup V'_{\text{shared}})\right)\right|$$

Besides, $W \setminus (S \cup V'_{\text{shared}}) = V'_{\text{opt}} \setminus X$ and, since S and V'_{shared} are disjoint:

$$\begin{aligned} \left|E\left(S \cup V'_{\text{shared}}, W \setminus (S \cup V'_{\text{shared}})\right)\right| &= \left|E\left(V'_{\text{shared}}, V'_{\text{opt}} \setminus X\right)\right| + \left|E\left(S, V'_{\text{opt}} \setminus X\right)\right| \\ &\geqslant \left|E\left(V'_{\text{shared}}, V'_{\text{opt}} \setminus X\right)\right| \end{aligned}$$

Recall that, by construction of S, $\delta(S \cup V'_{\text{shared}}) \leqslant \delta(X)$. Hence:

$$\delta(W) \leqslant \delta(X) + \delta\left(V'_{\text{opt}} \setminus X\right) - 2\left|E\left(V'_{\text{shared}}, V'_{\text{opt}} \setminus X\right)\right|$$

Also, by construction, $|E(V'', V'_{\text{opt}} \setminus X)| = 0$; so:

$$\begin{aligned} \delta(W) &\leqslant \delta(X) + \delta\left(V'_{\text{opt}} \setminus X\right) - 2\left(\left|E\left(V'_{\text{shared}}, V'_{\text{opt}} \setminus X\right)\right| + \left|E\left(V'', V'_{\text{opt}} \setminus X\right)\right|\right) \\ &= \delta(X) + \delta\left(V'_{\text{opt}} \setminus X\right) - 2\left|E\left(X, V'_{\text{opt}} \setminus X\right)\right| \;=\; \delta\left(V'_{\text{opt}}\right) \end{aligned} \quad (1)$$

where the first equality in (1) holds due to the fact V'_{shared} and V'' are disjoint.

By optimality of V'_{opt}, one gets $\delta(W) = \delta(V'_{\text{opt}})$. In the branching tree, we move to the one son taking S and we follow any branch until our algorithm deviates again from the optimal solution. Then we iterate the same hybridation trick. After at most k steps, we get to a leaf of the branching tree with a solution V' which is as good as the optimal solution. □

Let us note that Algorithm `MINCUT`(G,T,R,k) can be slightly modified in order to work also for another fixed-cardinality problem, the k-DENSEST SUBGRAPH problem, which consists in determining a subset of k vertices maximizing the number of edges in the subgraph induced by them. Indeed, it suffices to switch from minimization of the number of outgoing edges to maximization of the number of inner edges. The rest of the proof remains the same.

Corollary 6. *k-DENSEST SUBGRAPH, parameterized by $k\Delta$, is FPT.*

We now return to the standard parameterization of MIN $(k, n-k)$-CUT. Unfortunately, unlike what have been done for MAX $(k, n-k)$-CUT, we have not been able to show until now that the case $p < k$ is "trivial". However, we can prove that when $p \geqslant k$, then MIN $(k, n-k)$-CUT parameterized by the value p of the solution is FPT. This is an immediate corollary of the following proposition.

Proposition 7. MIN $(k, n-k)$-CUT *parameterized by $p + k$ is FPT.*

Proof. Each vertex v such that $|N(v)| \geqslant k + p$ has to be in $V \setminus V'$ (of size $n - k$). Indeed, if one puts v in V' (of size k), among its $k + p$ incident edges, at least $p + 1$ leave from V'; so, it cannot yield a feasible solution. All the vertices v such that $|N(v)| \geqslant k + p$ are then rejected. Thus, one can adapt the FPT algorithm in $k\Delta$ of Theorem 4 by considering the k-neighborhood of a vertex v not in the whole graph G, but in $G[T \cup U]$. One can easily check that the algorithm still works and since in those subgraphs the degree is bounded by $p+k$ we get an FPT algorithm in $p+k$. □

In [8], it is shown that, for any $\varepsilon > 0$, there exists a randomized $(1 + \frac{\varepsilon k}{\log n})$-approximation for MIN $(k, n-k)$-CUT. From this result, we can easily derive that when $p < \frac{\log n}{k}$ then the problem is solvable in polynomial time (by a randomized algorithm). Indeed, fixing $\varepsilon = 1$, the algorithm in [8] is a $(1 + \frac{k}{\log(n)})$-approximation. This approximation ratio is strictly better than $1 + \frac{1}{p}$. This means that the algorithm outputs a solution of value lower than $p + 1$, hence at most p, if there exists a solution of value at most p.

We now conclude this section by showing that, when $p \leqslant k$, MIN $(k, n-k)$-CUT can be solved in time $O^*(n^p)$.

Proposition 8. *If $p \leqslant k$, then* MIN $(k, n-k)$-CUT *can be solved in time $O^*(n^p)$.*

Proof. Since $p \leqslant k$, there exist in the optimal set V', $p' \leqslant p$ vertices incident to the p outgoing edges. So, the $k-p'$ remaining vertices of V' induce a subgraph that is disconnected from $G[V \setminus V']$.

Hence, one can enumerate all the $p' \leqslant p$ subsets of V. For each such subset $\widetilde{V}$, the graph $G[V \setminus \widetilde{V}]$ is disconnected. Denote by $C = (C_i)_{0 \leqslant i \leqslant |C|}$ the connected components of $G[V \setminus \widetilde{V}]$ and by α_i the number of edges between C_i and $\widetilde{V}$. We have to pick a subset $C' \subset C$ among these components such that $\sum_{C_i \in C'} |C_i| = k - p'$ and maximizing $\sum_{C_i \in C'} \alpha_i$. This can be done in polynomial time using standard dynamic programming techniques. □

3 Parameterization by k and approximation

Recall that both MAX and MIN $(k, n-k)$-CUT parameterized by k are W[1]-hard [6, 3]. In this section, we give some approximation algorithms working in FPT time with respect to parameter k.

Proposition 9. MAX $(k, n-k)$-CUT*, parameterized by k has a fixed-parameter approximation schema.*

Proof. Fix some $\varepsilon > 0$. Given a graph $G = (V, E)$, let $d_1 \leqslant d_2 \leqslant \ldots \leqslant d_k$ be the degrees of the k largest-degree vertices $v_1, v_2, \ldots v_k$ in G. An optima solution of value opt is obviously bounded above by $B = \Sigma_{i=1}^{k} d_i$. Now, consider solution $V' = \{v_1, v_2, \ldots, v_k\}$. As there exist at most $k(k-1)/2 \leqslant k^2/2$ (when V' is a k-clique) inner edges, solution V' has a value sol at least $B - k^2$. Hence, the approximation ratio is at least $\frac{B-k^2}{B} = 1 - \frac{k^2}{B}$. Since, obviously, $B \geqslant d_1 = \Delta$, an approximation ratio at least $1 - \frac{k^2}{\Delta}$ is immediately derived.

If $\varepsilon \geqslant \frac{k^2}{\Delta}$ then V' is a $(1-\varepsilon)$-approximation. Otherwise, if $\varepsilon \leqslant \frac{k^2}{\Delta}$, then $\Delta \leqslant \frac{k^2}{\varepsilon}$. So, the branching algorithm of Theorem 3 with time-complexity $O^*(\Delta^k)$ is in this case an $O^*(\frac{k^{2k}}{\varepsilon^k})$-time algorithm. □

Proposition 10. MIN $(k, n-k)$-CUT *parameterized by k has a randomized fixed-parameter approximation schema.*

Proof. For any $\varepsilon > 0$, if $k < \log n$, then according to the result of [8], there exists a randomized polynomial time $(1+\varepsilon)$-approximation. Else, if $k > \log n$, the exhaustive enumeration of the k-subsets takes time $O^*(n^k) = O^*((2^k)^k) = O^*(2^{k^2})$. □

Finding approximation algorithms that work in FPT time with respect to parameter p is an interesting question. Combining the result of [8] and an $O(\log^{1.5}(n))$-approximation algorithm in [9] we can show that the problem is $O(k^{3/5})$ approximable in polynomial time by a randomized algorithm.

We now show that an approximation ratio $\frac{k^2}{f(k)} + 1$ for MIN $(k, n-k)$-CUT can be achieved in time $O^*(n^{f(k)})$. This, for instance, concludes a ratio $o(k^2)$ in time $O^*(n^{o(k)})$. We distinguish three cases with respect to the parameter p. If $p \geqslant k$, then by the discussion just above, since any solution has size at most $k(k+p)$, an approximation ratio at most $2k$ is immediately derived. Assume now $p \leqslant k$. Here, we distinguish two subcases, namely $p \leqslant f(k)$ and $k \geqslant p \geqslant f(k)$. In the first of the subcases, using Proposition 8, an optimal solution for MIN $(k, n-k)$-CUT can be found in time at most $O^*(n^{f(k)})$. For the second subcase, consider a solution consisting of taking the set V' of the k vertices of G with lowest degrees, and denote by σ the sum of these degrees. Then,

the value opt of an optimal solution is at least $\sigma - k^2$, i.e., $\sigma \leqslant \text{opt} + k^2$. Hence, if $p < \sigma - k^2$, the algorithm answers "no"; otherwise, some easy algebra leads to an approximation ratio bounded above by $\frac{k^2}{f(k)} + 1$.

We feel however that much better results should be achievable for MIN $(k, (n-k))$-CUT in FPT time.

4 Other parameterizations

When dealing with parameterization of graph problems, some classical parameters arise naturally. One of them, very frequently used in the fixed parameter literature is the treewidth of the graph.

In [2], an algorithm solving the k-DENSEST problem in $O^*(2^{\text{tw}})$ is given. We show here that this algorithm could be adapted for a large class of graph problems including MIN and MAX $(k, n-k)$-CUT.

Definition 11*:* A local graph partitioning problem is a problem having as input a graph $G = (V, E)$ and two integers k and p. Feasible solutions are subsets $V' \subseteq V$ of size k. The value of a solution, denoted by $\text{val}(V')$, is a linear combination $\alpha_1 m_1 + \alpha_2 m_2$ where $m_1 = |E(V')|$, $m_2 = |E(V', V \setminus V')|$ and $\alpha_1, \alpha_2 \in \mathbb{R}$. The goal is to determine whether there exists a solution of value at least p (for a maximization problem) or at most p (for a minimization problem). □

Note that $\alpha_1 = 1$, $\alpha_2 = 0$ corresponds to k-DENSEST SUBGRAPH and k-SPARSEST SUBGRAPH, while $\alpha_1 = 0$, $\alpha_2 = 1$ corresponds $(k, n-k)$-CUT, and $\alpha_1 = \alpha_2 = 1$ gives k-COVERAGE. Not only any fixed cardinality problem (as those mentioned above and in [3]) but many other problems fit Definition 11. For instance, a very well-known problem, the MIN BISECTION problem where one wishes to minimize the cut between the sets of an equipartition of the vertices into two sets, is also a local graph partitioning problem.

Proposition 12. *Any local graph partitioning problem can be solved in time* $O^*(2^{\text{tw}})$.

Proof. A tree decomposition of a graph $G(V, E)$ is a pair (X, T) where T is a tree on vertex set $N(T)$ the vertices of which are called nodes and $X = (\{X_i : i \in N(T)\})$ is a collection of subsets of V such that: (i) $\cup_{i \in N(T)} X_i = V$, (ii) for each edge $(v, w) \in E$, there exist an $i \in N(T)$ such that $\{v, w\} \in X_i$, and (iii) for each $v \in V$, the set of nodes $\{i : v \in X_i\}$ forms a subtree of T. The width of a tree decomposition $(\{X_i : i \in N(T)\}, T)$ equals $\max_{i \in N(T)}\{|X_i| - 1\}$. The treewidth of a graph G is the minimum width over all tree decompositions of G. We say that a tree decomposition is nice if any node of its tree that is not the root is one of the following types:

- a leaf that contains a single vertex from the graph;
- an introduce node X_i with one child X_j such that $X_i = X_j \cup \{v\}$ for some vertex $v \in V$;
- a forget node X_i with one child X_j such that $X_j = X_i \cup \{v\}$ for some vertex $v \in V$;
- a join node X_i with two children X_j and X_l such that $X_i = X_j = X_l$.

Assume that the local graph partitioning problem Π is a minimization problem (we want to find V' such that $\text{val}(V') \leqslant p$), the maximization case being similar. An algorithm that transforms in linear time an arbitrary tree decomposition into a nice one with the same treewidth is presented in [10]. Consider a nice tree decomposition of G and let T_i be the subtree of T rooted at X_i, and $G_i = (V_i, E_i)$ be the subgraph of G induced by the vertices in $\bigcup_{X_j \in T_i} X_j$. For each node $X_i = (v_1, v_2, \ldots, v_{|X_i|})$ of the tree decomposition, define a configuration vector $\vec{c} \in \{0,1\}^{|X_i|}$; $\vec{c}[j] = 1 \iff v_j \in X_i$ belongs to the solution. Moreover, for each node X_i, consider a table A_i of size $2^{|X_i|} \times (k+1)$. Each row of A_i represents a configuration and each column represents the number k', $0 \leqslant k' \leqslant k$, of vertices in $V_i \setminus X_i$ included in the solution. The value of an entry of this table equals the value of the best solution respecting both the configuration vector and the number k', and $-\infty$ is used to define an infeasible solution. In the sequel, we set $X_{i,t} = \{v_h \in X_i : \vec{c}(h) = 1\}$ and $X_{i,r} = \{v_h \in X_i : \vec{c}(h) = 0\}$.

The algorithm examines the nodes of T in a bottom-up way and fills in the table A_i for each node X_i. In the initialization step, for each leaf node X_i and each configuration $\vec{c}$, we have $A_i[\vec{c}, k'] = 0$ if $k' = 0$; otherwise $A_i[\vec{c}, k'] = -\infty$.

If X_i is a forget node, then consider a configuration $\vec{c}$ for X_i. In X_j this configuration is extended with the decision whether vertex v is included into the solution or not. Hence, taking into account that $v \in V_i \setminus X_i$ we get:

$$A_i\left[\vec{c}, k'\right] = \min\left\{A_j\left[\vec{c} \times \{0\}, k'\right], A_j\left[\vec{c} \times \{1\}, k' - 1\right]\right\}$$

for each configuration $\vec{c}$ and each k', $0 \leqslant k' \leqslant k$.

If X_i is an introduce node, then consider a configuration $\vec{c}$ for X_j. If v is taken in V', its inclusion adds the quantity $\delta_v = \alpha_1|E(\{v\}, X_{i,t})| + \alpha_2|E(\{v\}, X_{i,r})|$ to the solution. The crucial point is that δ_v does not depend on the k' vertices of $V_i \setminus X_i$ taken in the solution. Indeed, by construction a vertex in $V_i \setminus X_i$ has its subtree entirely contained in T_i. Besides, the subtree of v intersects T_i only in its root, since v appears in X_i, disappears from X_j and has, by definition, a connected subtree. So, we know that there is no edge in G between v and any vertex of $V_i \setminus X_i$. Hence, $A_i[\vec{c} \times \{1\}, k'] = A_j[\vec{c}, k'] + \delta_v$, since k' counts only the vertices of the current solution in $V_i \setminus X_i$. The case where v is discarded from the solution (not taken in V') is completely similar; we just define δ_v according to the number of edges linking v to vertices of T_i respectively in V' and not in V'.

If X_i is a join node, then for each configuration $\vec{c}$ for X_i and each k', $0 \leqslant k' \leqslant k$, we have to find the best solution obtained by k_j, $0 \leqslant k_j \leqslant k'$, vertices in A_j plus $k' - k_j$ vertices in A_l. However, the quantity $\delta_{\vec{c}} = \alpha_1|E(X_{i,t})| + \alpha_2|E(X_{i,t}, X_{i,r})|$ is counted twice. Note that $\delta_{\vec{c}}$ depends only on $X_{i,t}$ and $X_{i,r}$, since there is no edge between $V_l \setminus X_i$ and $V_j \setminus X_i$. Hence, we get:

$$A_i\left[\vec{c}, k'\right] = \max_{0 \leqslant k_j \leqslant k'} \left\{A_j\left[\vec{c}, k_j\right] + A_l\left[\vec{c}, k' - k_j\right]\right\} - \delta_c$$

and the proof of the proposition is completed. □

Corollary 13. MAX *and* MIN $(k, n-k)$-CUT *parameterized by the treewidth of the input graph are FPT.*

Corollary 14. *Restricted to trees,* MAX *and* MIN $(k, n-k)$-CUT *can be solved in polynomial time.*

Corollary 15. MIN BISECTION *parameterized by the treewidth of the input graph is FPT.*

It is worth noticing that the result easily extends to the weighted case (edges have weight) and to the case of partitioning V into a constant number of classes (with a higher running time).

Another natural parameter frequently used in the parameterized complexity framework is the size τ of a minimum vertex cover of the input graph. Since it always holds that $\mathrm{tw} \leqslant \tau$, the result of Proposition 12 immediately applies to parameterization by τ. However, the algorithm developed there needs exponential space. In what follows, we give a parameterization by τ using polynomial space.

Proposition 16. MAX *and* MIN $(k, n-k)$-CUT *parameterized by* τ *can be solved in FPT* $O^*(2^\tau)$ *time and in polynomial space.*

Proof. Consider the following algorithm:

- compute a minimum vertex cover C of G;
- for every subset X of C of size $|X|$ smaller than k, complete X with the $k - |X|$ vertices of $V \setminus C$ that maximize (resp., minimize) their incidence with $C \setminus X$ (i.e., the number of neighbours in $C \setminus X$);
- output the best solution.

Recall that a minimum size vertex cover can be computed in time $O^*(1.2738^\tau)$ time by means of the fixed-parameter algorithm of [4] and using polynomial space. The operation on every subset is polynomial, so the global computation time is at most $O^*(2^\tau)$.

The soundness follows from the fact that a complement of a vertex cover is an independent set. Denoting by V' the optimal vertex-set (i.e., the k vertices inducing an optimal cut), then $V' \cap C$ will be considered by the above algorithm, and then every vertex of the completion will add exactly to the solution its number of neighbors in $V' \cap C$, which is maximized (or minimized) in the algorithm. □

5 Concluding remarks

Table 1 sums up results about parameterizations of MAX and MIN $(k, n-k)$-CUT. The first two lines concern exact algorithms, while the last two deal with approximations. The bold entries show our contribution. Notatios fpt(r)AS denote FPT (random) approximation schema. Let us note that our results derive some additional interesting corollaries, for instance, *both problems studied, when parameterized by k are FPT in graphs of bounded degree.*

Parameters	k	$k\Delta$	p	τ	tw
MAX $(k, n-k)$-CUT	W[1]-hard	**FPT**	**FPT**	**FPT**	**FPT**
MIN $(k, n-k)$-CUT	W[1]-hard	**FPT**	**FPT for p > k**	**FPT**	**FPT**
MAX $(k, n-k)$-CUT	**fptAS**				
MIN $(k, n-k)$-CUT	**fptrAS**				

Table 1: The state of the art of MAX and MIN $(k, n-k)$-CUT.

Settlement of the parameterized complexity of MIN $(k, n-k)$-CUT with respect to p is, in our opinion, the major open problem deserving further efforts. Note that, if this problem is FPT, then this result would immediately apply to the case of MIN BISECTION, whose standard parameterization is an open problem. Another interesting open question, in the case where MIN $(k, n-k)$-CUT is not FPT, is to design parameterized (with respect to p) approximation algorithms achieving ratios better than $O(k^2)$ (deterministic algorithm) and $O(k^{3/5})$ (randomized algorithms).

Finally, we hope that applying the idea of devising search tree based on the exploration of the neighborhood of a greedily chosen vertex could be fruitfully applied to other problems.

References

[1] A. A. Ageev and M. Sviridenko. Approximation algorithms for maximum coverage and max cut with given sizes of parts. In G. Cornuéjols, R. E. Burkard and G. J. Woeginger, editors, *Proc. of the Conference on Integer Programming and Combinatorial Optimization, IPCO'99*, volume 1610 of *Lecture Notes in Computer Science*, pp. 17–30. Springer, 1999.

[2] N. Bourgeois, A. Giannakos, G. Lucarelli, I. Milis, and V. Th. Paschos. Exact and approximation algorithms for DENSEST k-SUBGRAPH. Cahier du LAMSADE 324, LAMSADE, Université Paris-Dauphine, 2012.

[3] L. Cai. Parameter complexity of cardinality constrained optimization problems. *The Computer Journal*, 51:102–121, 2008.

[4] J. Chen, I. A. Kanj, and G. Xia. Improved upper bounds for vertex cover. *Theoret. Comput. Sci.*, 411(40-42):3736–3756, 2010.

[5] B. Courcelle. The monadic second-order logic of graphs. i. recognizable sets of finite graphs. *Information and Computation*, 85:12–75, 1990.

[6] R. G. Downey, V. Estivill-Castro, M. R. Fellows, E. Prieto, and F. A. Rosamond. Cutting up is hard to do: the parameterized complexity of k-cut and related problems. In *Electronic Notes in Theoretical Computer Science 78*, pages 205–218. Elsevier, 2003.

[7] R. G. Downey and M. R. Fellows. *Parameterized complexity*. Monographs in Computer Science. Springer, New York, 1999.

[8] U. Feige, R. Krauthgamer, and K. Nissim. On cutting a few vertices from a graph. *Discrete Appl. Math.*, 127(3):643–649, 2003.

[9] U. Feige and M. Langberg. Approximation algorithms for maximization problems arising in graph partitioning. *J. Algorithms*, 41(2):174–211, 2001.

[10] T. Kloks. *Treewidth, computations and approximations*, volume 842 of *Lecture Notes in Computer Science*. Springer, 1994.

[11] H. R. Lewis and C. H. Papadimitriou. *Elements of the theory of computation*. Prentice-Hall, 1981.

[12] S. Maneth. Logic and automata. Lecture 3: Expressiveness of MSO graph properties. Logic Summer School, December 2006.

[13] D. Marx. Parameterized complexity and approximation algorithms. *The Computer Journal*, 51(1):60–78, 2008.

[14] S. Szeider. Monadic second order logic on graphs with local cardinality constraints. In E. Ochmański and J. Tyszkiewicz, editors, *Proc. Mathematical Foundations of Computer Science, MFCS'08*, volume 5162 of *Lecture Notes in Computer Science*, pages 601–612. Springer, 2008.