Multi-parameter Analysis for Local Graph Partitioning Problems: Using Greediness for Parameterization*

Édouard Bonnet[1], Bruno Escoffier[2,3], Vangelis Th. Paschos[1**], and Émeric Tourniaire[1]

[1] PSL Research University, Université Paris-Dauphine, LAMSADE, CNRS UMR 7243
{edouard.bonnet,paschos,emeric.tourniaire}@lamsade.dauphine.fr
[2] Sorbonne Universités, UPMC Université Paris 06, UMR 7606, LIP6, F-75005, Paris, France
[3] CNRS, UMR 7606, LIP6, F-75005, Paris, France
bruno.escoffier@lip6.fr

Abstract. We study the parameterized complexity of a broad class of problems called "local graph partitioning problems" that includes the classical fixed cardinality problems as MAX k-VERTEX COVER, k-DENSEST SUBGRAPH, etc. By developing a technique that we call "greediness-for-parameterization", we obtain fixed parameter algorithms with respect to a pair of parameters k, the size of the solution (but *not* its value) and Δ, the maximum degree of the input graph. In particular, greediness-for-parameterization improves asymptotic running times for these problems upon random separation (that is a special case of color coding) and is more intuitive and simple. Then, we show how these results can be easily extended for getting standard-parameterization results (i.e., with parameter the value of the optimal solution) for a well known local graph partitioning problem.

1 Introduction

A local graph partitioning problem is a problem defined on some graph $G = (V, E)$ with two integers k and p. Feasible solutions are subsets $V' \subseteq V$ of size exactly k. The value of their solutions is a linear combination of sizes of edge-subsets and the objective is to determine whether there exists a solution of value at least or at most p. Problems as MAX k-VERTEX COVER, MIN k-VERTEX COVER, k-DENSEST SUBGRAPH, k-SPARSEST SUBGRAPH, MAX $(k, n-k)$-CUT and MIN $(k, n-k)$-CUT, also known as fixed cardinality problems, are local graph partitioning problems. When dealing with graph problems, several natural parameters, other than the size p of the optimum, can be of interest, for instance, the maximum degree Δ of the input graph, its treewidth, etc. To these parameters, common for any graph problem, in the case of local graph partitioning problem handled here, one more natural parameter of great interest can be additionally considered, the size k of V'. For instance, most of these problems have mainly been studied in [4, 9], from a parameterized

* Research supported by the French Agency for Research under the program TODO, ANR-09-EMER-010

** Also, Institut Universitaire de France

point of view, with respect to parameter k, and have been proven W[1]-hard. Dealing with standard parameterization, the only problems that, to the best of our knowledge, have not been studied yet, are the MAX $(k, n-k)$-CUT and the MIN $(k, n-k)$-CUT problems.

In this paper we develop a technique for obtaining multi-parameterized results for local graph partitioning problems. Informally, the basic idea behind it is the following. Perform a branching with respect to a vertex chosen upon some greedy criterion. For instance, this criterion could be to consider some vertex v that maximizes the number of edges added to the solution under construction. Without branching, such a greedy criterion is not optimal. However, if at each step either the greedily chosen vertex v, or some of its neighbors (more precisely, a vertex at bounded distance from v) are a good choice (they are in an optimal solution), then a branching rule on neighbors of v leads to a branching tree whose size is bounded by a function of k and Δ, and at least one leaf of which is an optimal solution. This method, called "greediness-for-parameterization", is presented in Section 2 together with interesting corollaries about particular local graph partitioning problems.

The results of Section 2 can sometimes be easily extended to standard parameterization results. In Section 3 we study standard parameterization of the two still unstudied fixed cardinality problems MAX and MIN $(k, n-k)$-CUT. We prove that the former is fixed parameter tractable (FPT), while, unfortunately, the status of the latter one remains still unclear. In order to handle MAX $(k, n-k)$-CUT we first show that when $p \leqslant k$ or $p \leqslant \Delta$, the problem can be solved in polynomial time. So, the only "non-trivial" case occurs when $p > k$ and $p > \Delta$. That case is handled by greediness-for-parameterization. Unfortunately, this method concludes inclusion of MIN $(k, n-k)$-CUT in FPT only for some particular cases. Note that in a very recent technical report of Fomin et al. [12], the following problem is considered: given a graph G and two integers k, p, determine whether there exists a set $V' \subset V$ of size *at most* k such that at most p edges have exactly one endpoint in V'. They prove that this problem is FPT with respect to p. Let us underline the fact that looking for a set of size at most k seems to be radically different from looking for a set of size exactly k (as in MIN $(k, n-k)$-CUT). For instance, in the case $k = n/2$, the former becomes the MIN CUT problem that is in P, while the latter becomes the MIN BISECTION problem that is NP-hard.

In Section 4.1, 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 with ratio $g(.)$ for Π is an algorithm running in time $f(\ell)|I|^{O(1)}$ that either finds an approximate solution of value at least/at most $g(\ell)\ell$, or reports that there is no solution of value at least/at most 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, where one has to find the subset of k vertices which cover the greatest number of edges [17]. Note that the existence of problems having this behaviour but with respect to the standard parameter is an open question in [17]. Let us note that polynomial approximation of MIN $(k, n-k)$-CUT has been studied in [10] where it is proven 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 [11]), for all k.

Finally, in Section 4.2, we handle parameterization of local graph partitioning problems by the treewidth tw of the input graph and show, using a standard dynamic programming technique, that they admit an $O^*(2^{\text{tw}})$-time FPT algorithm, where the $O^*(\cdot)$ notation ignores polynomial factors. 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 some local partitioning problems (this is the case, for instance, of MAX and MIN $(k, n-k)$-CUT) do not fit Courcelle's Theorem [7]. 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 cardinality 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, [15]) that E is not recognizable [16], a contradiction. Hence, MAX and MIN $(k, n-k)$-CUT are not expressible in MSO; consequently, the fact that those two problems, parameterized by treewidth (tw) are FPT cannot be obtained by Courcelle's Theorem. Furthermore, even several known extended variants of MSO which capture more problems [19], do not seem to be able to express the equality of two sets either.

2 Greediness-for-parameterization

We first formally define the class of local graph paritioning problems. We use the standard notation to deal with graphs. If $G = (V, E)$ is a graph and $X \subseteq V$ is a subset of vertices, $E(X)$ is the set of edges in the subgraph $G[X]$ induced by X, and $E(X, Y)$ is the set of edges having one endpoint in X and one endpoint in Y.

Definition 1. *In a local graph partitioning problem, given a graph $G = (V, E)$ and two integers k and p, one has to find a subset V' of size exactly k such that the value of the solution* $\text{val}(V') \stackrel{def}{=} \alpha_1|E(V')| + \alpha_2|E(V', V \setminus V')| \geqslant p$, *(resp., $\leqslant p$ for a minimization problem) where α_1 and α_2 are real constants.*

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 to $(k, n-k)$-CUT, and $\alpha_1 = \alpha_2 = 1$ gives k-VERTEX COVER. As a local graph partitioning problem is entirely defined by α_1, α_2 and goal $\in \{\min, \max\}$ we will unambiguously denote by $\mathcal{L}(\text{goal}, \alpha_1, \alpha_2)$ the corresponding problem. For conciseness and when no confusion is possible, we will use *local problem* instead. In the sequel, k always denotes the size of feasible subset of vertices and p the standard parameter, i.e., the solution-size. Moreover, as a partition into k and $n-k$ vertices, respectively, is completely defined by the subset V' of size k, we will consider it to be the solution. A *partial solution* T is a subset of V' with less than k vertices. Similarly to the value of a solution, we define the value of a partial solution, and denote it by $\text{val}(T)$.

Informally, we devise algorithms for local problems that add vertices to an initially empty set T (for "taken" vertices) and stop when T becomes of size k, i.e., when T itself becomes a feasible solution. A vertex introduced in T is irrevocably introduced there and will be not removed later.

Definition 2. *Given a local graph partitioning problem* $\mathcal{L}(\text{goal}, \alpha_1, \alpha_2)$*, the* contribution *of a vertex* v *within a partial solution* T *(such that* $v \in T$*) is defined by* $\delta(v, T) = \frac{1}{2}\alpha_1 |E(\{v\}, T)| + \alpha_2 |E(\{v\}, V \setminus T)|$.

Note that the value of any (partial) solution T satifies $\text{val}(T) = \Sigma_{v \in T} \delta(v, T)$. One can also remark that $\delta(v, T) = \delta(v, T \cap N(v))$, where $N(v)$ denotes the (open) neighbourhood of the vertex v. Function δ is called the *contribution function* or simply the *contribution* of the corresponding local problem. We introduce now the notion of *degrading* contribution. Intuitively, a contribution is degrading if, while vertices are added to the solution, the contribution of a vertex already added to the solution can only decrease.

Definition 3. *Given a local graph partitioning problem* $\mathcal{L}(\text{goal}, \alpha_1, \alpha_2)$*, a contribution function is said to be* degrading *if for every* v*,* T *and* T' *such that* $v \in T \subseteq T'$*,* $\delta(v, T) \leqslant \delta(v, T')$ *for* goal $= \min$ *(resp.,* $\delta(v, T) \geqslant \delta(v, T')$ *for* goal $= \max$*).*

Note that it can be easily shown that for a maximization problem, a contribution function is degrading if and only if $\alpha_2 \geqslant \alpha_1/2$ ($\alpha_2 \leqslant \alpha_1/2$ for a minimization problem). So in particular MAX k-VERTEX COVER, k-SPARSEST SUBGRAPH and MAX $(k, n-k)$-CUT have a degrading contribution function.

Theorem 1. *Every local partitioning problem having a degrading contribution function can be solved in time* $O^*(\Delta^k)$.

Proof. With no loss of generality, we carry out the proof for a minimization local problem $\mathcal{L}(\min, \alpha_1, \alpha_2)$. We recall that T will be a partial solution and eventually a feasible solution. Consider the following algorithm $\texttt{DLGPP}(T, k)$ which branches upon the closed neighborhood $N[v]$ of a vertex v minimizing the greedy criterion $\delta(v, T \cup \{v\})$:

Algorithm 1: A description of the algorithm DLGPP.

Input: A graph $G = (V, E)$, an integer k, and a triple goal $\in \{\min, \max\}$, α_1, α_2 defining a degrading local graph partioning problem $\mathcal{L}(\text{goal}, \alpha_1, \alpha_2)$.
Output: A set of k vertices $T \subseteq V$ optimizing val(T).
set $T = \emptyset$;
DLGPP(T, k):
if $k > 0$ **then**
  pick the vertex $v \in V \setminus T$ minimizing $\delta(v, T \cup \{v\})$;
  for *each vertex* $w \in N[v] \setminus T$ **do**
    run DLGPP($T \cup \{w\}$,$k-1$);
else
  ($k = 0$) store the feasible solution T;
return *the best among the solutions stored*;

The branching tree of DLGPP has depth k, since we add one vertex at each recursive call, and arity at most $\max_{v \in V} |N[v]| = \Delta + 1$, where $N[v]$ denotes the closed neighbourhood of v. Thus, the algorithm runs in $O^*(\Delta^k)$.

For the optimality proof, we use a classical hybridation technique between some optimal solution and the one solution computed by DLGPP.

Consider an optimal solution V'_{opt} different from the solution V' computed by DLGPP. A node s of the branching tree has two characteristics: the partial solution $T(s)$ at this node (denoted simply T if no ambiguity occurs) and the vertex chosen by the greedy criterion $v(s)$ (or simply v). We say that a node s of the branching tree *conforms* with the optimal solution V'_{opt} if $T(s) \subseteq V'_{\text{opt}}$. A node s *deviates* from the optimal solution V'_{opt} if none of its sons conforms with V'_{opt}.

We start from the root of the branching tree and, while possible, we move to a conform son of the current node. At some point we reach a node s which deviates from V'_{opt}. We set $T = T(s)$ and $v = v(s)$. Intuitively, T corresponds to the shared choices between the optimal solution and DLGPP made along the branch from the root to the node s of the branching tree. Setting $V_n = V'_{\text{opt}} \setminus T$, V_n does not intersect $N[v]$, otherwise s would not be deviating.

Choose any $z \in V_n$ and consider the solution induced by the set $V_e = V'_{\text{opt}} \cup \{v\} \setminus \{z\}$. We show that this solution is also optimal. Let $V_c = V'_{\text{opt}} \setminus \{z\}$. We have $\text{val}(V_e) = \Sigma_{w \in V_c} \delta(w, V_e) + \delta(v, V_e)$. Besides, $\delta(v, V_e) = \delta(v, V_e \cap N(v)) = \delta(v, T \cup \{v\})$ since $V_e \setminus (T \cup \{v\}) = V_n \setminus \{z\}$ and according to the last remark of the previous paragraph, $N(v) \cap V_n = \emptyset$. By the choice of v, $\delta(v, T \cup \{v\}) \leqslant \delta(z, T \cup \{z\})$, and, since δ is a degrading contribution, $\delta(z, T \cup \{z\}) \leqslant \delta(z, V'_{\text{opt}})$. Summing up, we get $\delta(v, V_e) \leqslant \delta(z, V'_{\text{opt}})$ and $\text{val}(V_e) \leqslant \Sigma_{w \in V_c} \delta(w, V_e) + \delta(z, V'_{\text{opt}})$. Since v is not in the neighborhood of V_n only z can degrade the contribution of those vertices, so $\Sigma_{w \in V_c} \delta(w, V_e) \leqslant \Sigma_{w \in V_c} \delta(w, V'_{\text{opt}})$, and $\text{val}(V_e) \leqslant \Sigma_{w \in V_c} \delta(w, V'_{\text{opt}}) + \delta(z, V'_{\text{opt}}) = \text{val}(V'_{\text{opt}})$.

Thus, by repeating this argument at most k times, we can conclude that the solution computed by DLGPP is as good as V'_{opt}. □

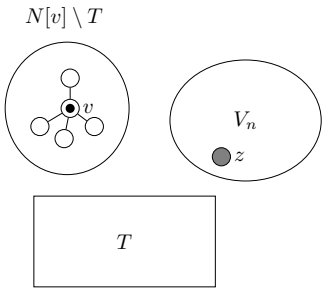


Fig. 1. Situation of the input graph at a deviating node of the branching tree. The vertex v can substitute z since, by the hypothesis, $N[v] \setminus T$ and V_n are disjoint and the contribution of a vertex can only decrease when we later add some of its neighbors in the solution.

Corollary 1. MAX k-VERTEX COVER, k-SPARSEST SUBGRAPH *and* MAX $(k, n-k)$-CUT *can be solved in* $O^*(\Delta^k)$.

As mentioned before, the local problems mentioned in Corollary 1 have a degrading contribution.

Theorem 2. *Every local partitioning problem can be solved in* $O^*((2k\sqrt{\Delta})^{2k})$.

Proof. Once again, with no loss of generality, we prove the theorem in the case of minimization, i.e., $\mathcal{L}(\min, \alpha_1, \alpha_2)$. Consider now the following algorithm LGPP(T, k):

Algorithm 2: A description of the algorithm LGPP.

Input: A graph $G = (V, E)$, an integer k, and a triple goal $\in \{\min, \max\}$, α_1, α_2 defining any local graph partioning problem $\mathcal{L}(\text{goal}, \alpha_1, \alpha_2)$.
Output: A set of k vertices $T \subseteq V$ optimizing val(T).
set $T = \emptyset$;
LGPP(T, k):
if $k > 0$ **then**
  for $i \leftarrow 1$ ***to*** k **do**
    find $S_i \subseteq V \setminus T$ minimizing $val(T \cup S_i)$, with S_i inducing a connected component of size i;
    for *each* $v \in S_i$ **do**
      run LGPP$(T \cup \{v\}, k-1)$;
  else
    $(k = 0)$ store the feasible solution T;
return *the best among the solutions stored*;

The proof of Theorem 2 involves an algorithm fairly similar to DLGPP but instead of branching on a vertex chosen greedily and its neighborhood, we branch on sets of vertices inducing connected components (also chosen greedily) and the neighborhood of those sets.

Let us first state the following straightforward lemma that bounds the number of induced connected components and the running time to enumerate them.

Lemma 1. *(Lemma 2 in [14]) The number of connected induced subgraphs of size bounded above by k is bounded by $n(4\Delta)^k$ and they can be enumerated in time $O^*((4\Delta)^k)$.*

In the branching tree of LGPP, a node has at most $k(k+1)/2 \leqslant k^2$ sons (i sons for S_i, i going from 1 to k). The depth of the branching tree is k. Hence, the size of the tree is $O(k^{2k})$. Computing the set S_i in each node takes time $O^*((4\Delta)^k)$ according to Lemma 1. Thus, the overall running-time of the algorithm is $O^*((2k\sqrt{\Delta})^{2k})$.

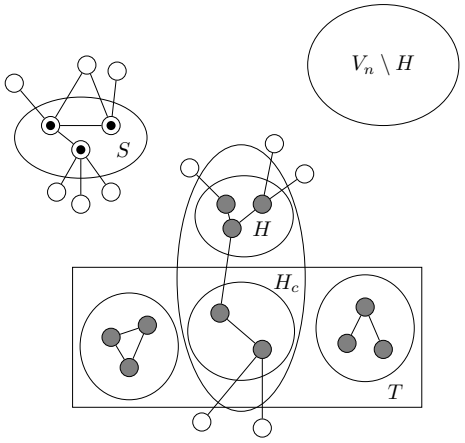


Fig. 2. Illustration of the proof, with filled vertices representing the optimal solution V'_{opt} and dotted vertices representing the set $S = S_{|H|}$ computed by LGPP which can substitute H, since V_n does not interact with H_c nor with S.

For the optimality of LGPP, we use the following lemma.

Lemma 2. *Let A,B,X, and Y be pairwise disjoint sets of vertices such that $\text{val}(A \cup X) \leqslant \text{val}(B \cup X)$, $N[A] \cap Y = \emptyset$ and $N[B] \cap Y = \emptyset$, where $N[X] = \bigcup_{v \in X} N[v]$. Then, $\text{val}(A \cup X \cup Y) \leqslant \text{val}(B \cup X \cup Y)$.*

Proof. Observe that $\text{val}(A \cup X \cup Y) = \text{val}(Y) + \text{val}(A \cup X) - 2\alpha_2|E(X,Y)| + \alpha_1|E(X,Y)| \leqslant \text{val}(Y) + \text{val}(B \cup X) - 2\alpha_2|E(X,Y)| + \alpha_1|E(X,Y)| = \text{val}(B \cup X \cup Y)$. □

We now show that LGPP is sound, using again hybridation between an optimal solution V'_{opt} and the one solution found by LGPP. We keep the same notation as in the proof of the soundness of DLGPP. Node s is a node of the branching tree which deviates from V'_{opt}, all nodes in the branch between the root and s conform with V'_{opt}, the shared choices constitute the set of vertices $T = T(s)$

and, for each i, set $S_i = S_i(s)$ (analogously to $v(s)$ in the previous proof, s is now linked to the subsets S_i computed at this node). Set $V_n = V'_{\text{opt}} \setminus T$. Take a maximal connected (non empty) subset H of V_n. Set $S = S_{|H|}$ and consider $V_e = V'_{\text{opt}} \setminus H \cup S = (T \cup V_n) \setminus H \cup S = T \cup S \cup (V_n \setminus H)$. Note that, by hypothesis, $N[S] \cap V_n = \emptyset$ since s is a deviating node. By the choice of S at the node s, $\text{val}(T \cup S) \leqslant \text{val}(T \cup H)$. So, $\text{val}(V_e) = \text{val}(T \cup S \cup (V_n \setminus H)) \leqslant \text{val}(T \cup H \cup (V_n \setminus H)) = \text{val}(T \cup V_n) = \text{val}(V'_{\text{opt}})$ according to Lemma 2, since by construction neither $N[H]$ nor $N[S]$, do intersect $V_n \setminus H$. Iterating the argument at most k times, we get to a leaf of the branching tree of `LGPP` which yields a solution as good as V'_{opt}. □

Corollary 2. *k-DENSEST SUBGRAPH and MIN $(k, n-k)$-CUT can be solved in time $O^*((2k\sqrt{\Delta})^{2k})$.*

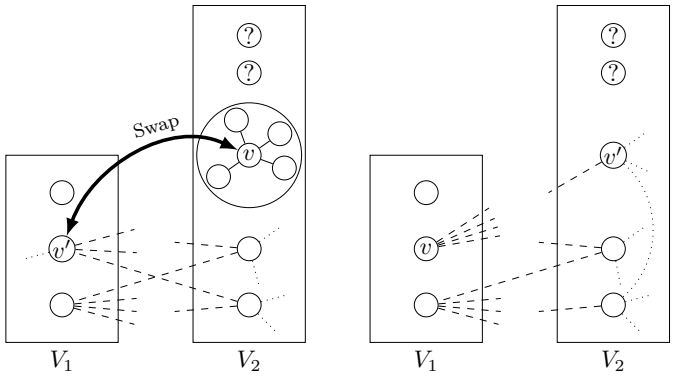
We simply observe, again, that the problems mentioned in Corollary 2 are local graph partitioning problems.

Theorem 1 improves the $O^*(2^{(\Delta+1)k}((\Delta+1)k)^{\log((\Delta+1)k)})$-time complexity for the corresponding problems given in [5] obtained there by the *random separation* technique, and Theorem 2 improves it whenever $k = 2^{o(\Delta)}$. Recall that random separation consists of randomly guessing if a vertex is in an optimal subset V' of size k (white vertices) or if it is in $N(V') \setminus V'$ (black vertices). For all other vertices the guess has no importance. As a right guess concerns at most only $k + k\Delta$ vertices, it is done with high probability if we repeat random guesses $f(k, \Delta)$ times with a suitable function f. Given a random guess, i.e., a random function $g : V \to \{\text{white,black}\}$, a solution can be computed in polynomial time by dynamic programming. Although random separation (and a fortiori *color coding* [2]) have also been applied to other problems than local graph partitioning ones, greediness-for-parameterization seems to be quite general and improves both running time and easiness of implementation since our algorithms do not need complex derandomizations.

Let us note that the greediness-for-parameterization technique can be even more general, by enhancing the scope of Definition 1 and can be applied to problems where the objective function takes into account not only edges but also vertices. The value of a solution could be defined as a function $\text{val} : \mathcal{P}(V) \to \mathbb{R}$ such that $\text{val}(\emptyset) = 0$, the contribution of a vertex v in a partial solution T is $\delta(v, T) = \text{val}(T \cup v) - \text{val}(T)$. Thus, for any subset T, $\text{val}(T) = \text{val}(T \setminus \{v_k\}) + \delta(v_k, T \setminus \{v_k\})$ where k is the size of T and v_k is the last vertex added to the solution. Hence, $\text{val}(T) = \Sigma_{1 \leqslant i \leqslant k} \delta(v_i, \{v_1, \ldots, v_{i-1}\}) + \text{val}(\emptyset) = \Sigma_{1 \leqslant i \leqslant k} \delta(v_i, \{v_1, \ldots, v_{i-1}\})$. Now, the only hypothesis we need to derive the same result as in Theorem 2 is the following: for each T' such that $(N(T') \setminus T) \cap (N(v) \setminus T) = \emptyset$, $\delta(v, T \cup T') = \delta(v, T)$.

Notice that under such generalization, MAX k-DOMINATING SET, asking for a set V' of k vertices that dominate the highest number of vertices in $V \setminus V'$ fulfills the enhancement just discussed. We therefore derive the following.

Corollary 3. *MAX k-DOMINATING SET can be solved in $O^*((2k\sqrt{\Delta})^{2k})$.*



(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.

Fig. 3. Illustration of a swapping

3 Standard Parameterization for MAX and MIN $(k, n-k)$-CUT

3.1 MAX $(k, n-k)$-CUT

In the sequel, we 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 Figure 3), 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 illustrated in Figure 4. The rest of the proof is an immediate application of Corollary 1.

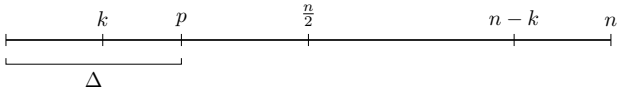


Fig. 4. Location of parameter p, relatively to k and Δ.

Lemma 3. *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 3). 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 more 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 $n-k$. □

Corollary 4. *In a graph with no isolated vertices, the optimal value for* MAX $(k, n-k)$-CUT *is at least* $\min\{n-k, k\}$.

Thus, on non-trivial instances, p is greater than k and Δ. Hence, Corollary 1 gives the following theorem.

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

3.2 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". So, Algorithm `LGPP` in Section 2 cannot be transformed into a standard FPT algorithm for this problem.

However, we can prove that if $p \geqslant k$, 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 1. 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. Let call U this set of vertices. Thus, one can adapt the FPT algorithm in $k+\Delta$ of Theorem 2 by considering the k-neighborhood of a vertex v not in the whole graph G, but in $G[V \setminus U]$. 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 [10], 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 [10] 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 proving that, when $p \leqslant k$, MIN $(k, n-k)$-CUT can be solved in time $O^*(n^p)$.

Proposition 2. *For $p \leqslant k$,* 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 subsets of V of size $p' \leqslant p$, and in particular, the subsets of size p' which disconnects the graph. For each such subset $\widetilde{V}$ such that the graph $G[V \setminus \widetilde{V}]$ is disconnected, we denote by $C = \{C_i\}_{1 \leqslant i \leqslant |C|}$ 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. □

4 Other Parameterizations

4.1 Parameterization by k and Approximation of $(k, n-k)$-CUT

Recall that both MAX and MIN $(k, n-k)$-CUT parameterized by k are W[1]-hard [9, 4]. In this section, we give some approximation algorithms working in FPT time with respect to parameter k.

Proposition 3. *When parameterized by k:*

- MAX $(k, n-k)$-CUT *admits a fixed-parameter approximation scheme;*
- MIN $(k, n-k)$-CUT *has a randomized fixed-parameter approximation scheme.*

Proof. We first handle MAX $(k, n-k)$-CUT. 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 optimal solution of value opt is obviously bounded from 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 $k(k-1)/2 \leqslant k^2$ inner edges (when V' is a k-clique), 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.

For MIN $(k, n-k)$-CUT, it is proven in [10] that, for $\varepsilon > 0$, if $k < \log n$, then there exists a randomized polynomial time $(1+\varepsilon)$-approximation. Else, if $k > \log n$, the exhaustive enumeration of the k-subsets takes time $O^*(n^k) = O^*((2^k)^k) = O^*(2^{k^2})$. □

We conclude this paragraph by showing 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 $O^*(n^{o(k)})$.

Proposition 4. *For every positive function f,* MIN $(k, n-k)$-CUT *is approximable within ratio $\frac{k^2}{f(k)}+1$ in time $O^*(n^{f(k)})$.*

Proof. We distinguish three cases with respect to the standard parameter p. If $p \geqslant k$, then an approximation ratio at most $2k$ is immediately derived, since any solution has size at most $k(k+p)$.

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 2, 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 $\frac{k^2}{f(k)}+1$. □

4.2 Parameterization by Treewidth and Vertex Cover Number

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.

It has already been proven that MIN and MAX $(k, n-k)$-CUT, as well as k-DENSEST SUBGRAPH can be solved in time $O^*(2^{\text{tw}})$ [3, 13]. We show here that the algorithm in [3] can be adapted to handle the whole class of local problems. 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 exists 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 [13].

Proposition 5. *Any local graph partitioning problem can be solved in time* $O^*(2^{\mathrm{tw}})$.

Proof. 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 $\boldsymbol{c} \in \{0,1\}^{|X_i|}$; $\boldsymbol{c}[j] = 1 \iff v_j \in X_i$ 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 $X_{i,t} = \{v_h \in X_i : \boldsymbol{c}(h) = 1\}$ and $X_{i,r} = \{v_h \in X_i : \boldsymbol{c}(h) = 0\}$.

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 $\boldsymbol{c}$, we have $A_i[\boldsymbol{c}, k'] = 0$ if $k' = 0$; otherwise $A_i[\boldsymbol{c}, k'] = -\infty$.

If X_i is a forget node, then consider a configuration $\boldsymbol{c}$ 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:

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

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

Assume X_i is an introduce node. Then consider a configuration $\boldsymbol{c}$ 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, $A_i[\boldsymbol{c} \times \{1\}, k'] = A_j[\boldsymbol{c}, k'] + \delta_v$, 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 $\boldsymbol{c}$ 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 $\delta_{\boldsymbol{c}} = \alpha_1|E(X_{i,t})| + \alpha_2|E(X_{i,t}, X_{i,r})|$ is counted twice. Note that $\delta_{\boldsymbol{c}}$ 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:

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

and the proof of the proposition is completed. □

Corollary 5. *Restricted to trees, any local graph partitioning problem can be solved in polynomial time.*

Corollary 6. 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 (where edges are weighted) 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 5 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 for a large number of local partitioning problems.

Proposition 6. *When parameterized by* τ, MAX *and* MIN k-VERTEX COVER, k-DENSEST SUBGRAPH *and* k-SPARSEST SUBGRAPH, MAX *and* MIN $(k, n-k)$-CUT *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) val (see Definition 1);
- 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 [6] 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 solution), then $V' \cap C$ will be considered by the above algorithm. For the problems handled, vertices from $V \setminus C$ can be ordered following a greedy criterion on their degrees and be chosen in order to complete $X = V' \cap C$ up to a set V' of k vertices that optimizes $\mathrm{val}(V')$. □

5 Open Questions

Of course, the main remaining open question is the parameterized complexity of MIN $(k, n-k)$-CUT with respect to the value of the solution p.

Another problem of interest is to look for a better algorithm for local graph partitioning problem in general. For instance, we can not rule out a time-complexity in $O^*((a\Delta)^{bk})$, with a and b two constants, which would be really closer to the $O^*(\Delta^k)$ complexity of the degrading contribution case.

Also, finding approximation algorithms that work in FPT time with respect to parameter p seems to be an equally interesting question. Combining the result of [10] and an $O(\log^{1.5}(n))$-approximation algorithm in [11], we can show that the problem is $O(k^{3/5})$ approximable in polynomial time by a randomized algorithm. But, is it possible to improve this ratio when allowing FPT time (with respect to p)?

Recent advances

We are glad to notice that, after the submission of this article and the presentation of the results at IPEC'13, two very interesting articles appeared on the same topic, and answer two of the open questions that we asked in the conclusion.

- In [8] it is shown that MIN $(k, n-k)$-CUT, parameterized by the value p of the solution, is solvable in time $O(2^{O(k^3)}n^3 \log^3 n)$. The technique used is completely different than the approach we have developed here.
- In [18], it is shown that a time-complexity of $O^*(4^{k+o(k)}\Delta^k)$ can be indeed reached for local graph partitioning problem in general.

References

1. Ageev, A.A., Sviridenko, M.: Approximation algorithms for maximum coverage and max cut with given sizes of parts. In Cornuéjols, G., Burkard, R.E., Woeginger, G.J., eds.: Proc. Conference on Integer Programming and Combinatorial Optimization, IPCO'99. Volume 1610 of Lecture Notes in Computer Science., Springer-Verlag (1999) 17–30
2. Alon, N., Yuster, R., Zwick, U.: Color-coding. J. Assoc. Comput. Mach. **42** (1995) 844–856
3. Bourgeois, N., Giannakos, A., Lucarelli, G., Milis, I., Paschos, V.: Exact and approximation algorithms for DENSEST k-SUBGRAPH. In Ghosh, S.K., Tokuyama, T., eds.: Proc. International Workshop on Algorithms and Computation, WALCOM'13. Volume 7748 of Lecture Notes in Computer Science., Springer-Verlag (2013) 114–125
4. Cai, L.: Parameter complexity of cardinality constrained optimization problems. The Computer Journal **51** (2008) 102–121
5. Cai, L., Chan, S.M., Chan, S.O.: Random separation: a new method for solving fixed-cardinality optimization problems. In Bodlaender, H.L., Langston, M.A., eds.: Proc. International Workshop on Parameterized and Exact Computation, IWPEC'06. Volume 4169 of Lecture Notes in Computer Science., Springer-Verlag (2006) 239–250
6. Chen, J., Kanj, I., Xia, G.: Improved upper bounds for vertex cover. Theoret. Comput. Sci. **411** (2010) 3736–3756
7. Courcelle, B.: The monadic second-order logic of graphs. i. recognizable sets of finite graphs. Information and Computation **85** (1990) 12–75
8. Cygan, M., Lokshtanov, D., Pilipczuk, M., Pilipczuk, M., Saurabh, S.: Minimum Bisection is fixed parameter tractable, CoRR, abs/1311.2563, 2013
9. 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. In: Electronic Notes in Theoretical Computer Science 78, Elsevier (2003) 205–218
10. Feige, U., Krauthgamer, R., Nissim, K.: On cutting a few vertices from a graph. Discrete Appl. Math. **127** (2003) 643–649
11. Feige, U., Langberg, M.: Approximation algorithms for maximization problems arising in graph partitioning. J. Algorithms **41** (2001) 174–211

12. Fomin, F.V., Golovach, P.A., Korhonen, J.H.: On the Parameterized Complexity of Cutting a Few Vertices from a Graph. Technical report, CoRR, abs/1304.6189 (2013)
13. Kloks, T.: Treewidth, computations and approximations. Volume 842 of Lecture Notes in Computer Science. Springer (1994)
14. Komusiewicz, C., Sorge, M.: Finding dense subgraphs of sparse graphs. In Thilikos, D., Woeginger, G., eds.: Proc. International Symposium on Parameterized and Exact Computation, IPEC'12. Volume 7535 of Lecture Notes in Computer Science., Springer-Verlag (2012) 242–251
15. Lewis, H.R., Papadimitriou, C.H.: Elements of the theory of computation. Prentice-Hall (1981)
16. Maneth, S.: Logic and automata. Lecture 3: Expressiveness of MSO graph properties. Logic Summer School (2006)
17. Marx, D.: Parameterized complexity and approximation algorithms. The Computer Journal **51** (2008) 60–78
18. Shachnai, H., Zehavi, M.:. Parameterized algorithms for graph partitioning problems To appear in the proceedings of WG'14.
19. Szeider, S.: Monadic second order logic on graphs with local cardinality constraints. In Ochmański, E., Tyszkiewicz, J., eds.: Proc. Mathematical Foundations of Computer Science, MFCS'08. Volume 5162 of Lecture Notes in Computer Science., Springer-Verlag (2008) 601–612