The Computer Journal Advance Access published November 3, 2007

doi:10.1093/comjnl/bxm086

Parameterized Complexity Of Cardinality Constrained Optimization Problems

Leizhen Cai*

Department of Computer Science and Engineering, The Chinese University of Hong Kong, Shatin, Hong Kong SAR, China
**Corresponding author: lcai@cse.cuhk.edu.hk*

We study the parameterized complexity of cardinality constrained optimization problems, i.e. optimization problems that require their solutions to contain specified numbers of elements to optimize solution values. For this purpose, we consider around 20 such optimization problems, as well as their parametric duals, that deal with various fundamental relations among vertices and edges in graphs. We have almost completely settled their parameterized complexity by giving either FPT algorithms or W[1]-hardness proofs. Furthermore, we obtain faster exact algorithms for several cardinality constrained optimization problems by transforming them into problems of finding maximum (minimum) weight triangles in weighted graphs.

Keywords: cardinality constrained optimization problem; exact algorithm; fixed-cardinality optimization problem; graph problem; parameterized complexity; FPT algorithm; W[1]-hardness

1. INTRODUCTION

1.1. Motivations

A cardinality constrained optimization problem requires its solutions to contain exactly a specified number of k elements to optimize solution values, and most combinatorial optimization problems have their natural cardinality constrained counterparts. For instance, the classical *vertex cover problem* asks for a minimum-size set of vertices in an n-vertex graph to cover all edges, and a cardinality constrained counterpart of it is the *maximum k-vertex cover problem* that requires us to find k vertices to cover as many edges as possible. We may also consider the *minimum k-vertex cover problem* that needs us to find k vertices to cover as few edges as possible. We note that the cardinality constrained optimization has a long history, and a recent survey article of Bruglieri *et al.* [1] gives an annotated bibliography of the subject, which has more than 100 references to various cardinality constrained optimization problems.

Normally, a cardinality constrained optimization problem is not easier than its corresponding optimization problem, and usually NP-hard when k is a part of input. On the other hand, it can usually be solved by exhaustive search in polynomial time for each fixed value of k, but the degree of the polynomial grows with k, which makes such algorithms neither useful in practice nor really meaningful in theory. This perhaps explains why most of the work in the literature on cardinality constrained optimization problems has been concentrated on approximation algorithms.

In this paper, we use the framework of Downey and Fellows [2] on parameterized complexity to study the cardinality constrained optimization problems. For this purpose, we regard such problems as parameterized problems by taking the cardinality of a solution (or a number related to it) as a fixed parameter k and refer to them as *fixed-cardinality optimization problems* to distinguish them from ordinary cardinality constrained optimization problems. Our main purpose is to design *FPT algorithms*, i.e. $f(k)|I|^{O(1)}$ time algorithms with $f(k)$ being a computable function of k and $|I|$ the size of input, to solve fixed-cardinality optimization problems, and, when such algorithms are unlikely to exist, establish the fixed-parameter intractability of the problems. In spite of extensive work on the parameterized complexity of numerous fixed-parameter problems and a long history of cardinality constrained optimization, the line of research in this paper is basically an unexplored area.[1] We note that, motivated by

[1]The author raised the issue of studying cardinality constrained optimization problems from the parameterized complexity point of view at the 1st Workshop on Parameterized Complexity held in Chennai, India, 2000, where he presented a W[1]-hardness proof for the maximum k-vertex cover problem. However, only a couple of isolated results [3, 4] have been reported since then.

the work in this paper, Cai *et al.* [5] have recently developed a powerful random separation method to solve fixed-cardinality optimization problems.

We will focus on fixed-cardinality optimization problems on graphs, especially fundamental problems that deal with various relations among vertices and edges. For around 20 graph problems, as well as their *parametric duals* (see the definition in Section 1.2), we almost completely settle their parameterized complexity by giving either FPT algorithms or proofs of fixed-parameter intractability. We also give faster exact algorithms for several fixed-parameter intractable ones by transforming them into problems of finding maximum (minimum) weight triangles in weighted graphs.

It is hoped that the work in this paper will show the usefulness and fruitfulness of parameterized complexity theory in studying cardinality constrained optimization problems, and also stir the reader's interest in pursuing further research in both parameterized complexity and cardinality constrained optimization.

1.2. Definitions and notation

Formally, a *cardinality constrained optimization problem* is a 4-tuple $(\mathcal{B}, \phi, k, opt)$, where $\mathcal{B}$ is a finite set called *solution base*, $\phi : 2^{\mathcal{B}} \to \{0, 1, 2, \ldots\} \cup \{-\infty, +\infty\}$ an objective function, k a non-negative integer and $opt \in \{min, max\}$. Our task is to find a *k-solution*, i.e. a subset $S \subseteq \mathcal{B}$ consisting of exactly k elements, to maximize (or minimize) the value $\phi(S)$. Note that, instead of specifying a set of feasible solutions, here we specify a feasible solution by its value, i.e. S is a feasible solution if $\phi(S) \neq -\infty$ for a maximization problem and $\phi(S) \neq +\infty$ for a minimization problem. If we require $\phi(S)$ to equal exactly a given number, then we have a *cardinality constrained exact-value problem*.

An instance of a *parameterized problem* consists of a pair (I, k) with I being the input and k the parameter. A *k-cardinality optimization problem* is a parameterized problem obtained from a cardinality constrained optimization problem by taking the solution size k as the parameter. The *parametric dual* of a k-cardinality optimization problem is a cardinality constrained optimization problem that requires us to find a solution $S \subseteq \mathcal{B}$ with exactly $|\mathcal{B}| - k$ elements, where k is the parameter, to maximize (or minimize) the value $\phi(S)$. For convenience, we sometimes use $\bar{k}$ for $|\mathcal{B}| - k$. Both k-cardinality optimization problems and their parametric duals are referred to as *fixed-cardinality optimization problems*. We can also define a *k-cardinality exact-value problem* and its parametric dual in a similar manner.

A parameterized problem (I, k) is *fixed-parameter tractable* (*FPT*) if it admits an FPT algorithm, i.e. an algorithm that runs in $f(k)|I|^{O(1)}$ time for some computable function $f(k)$ independent of the input size $|I|$. The class of FPT problems is denoted by FPT.

Downey and Fellows [2] have also introduced a W-hierarchy

$$W[1] \subseteq W[2] \subseteq W[3] \subseteq \cdots \subseteq W[XP]$$

to capture fixed-parameter intractability, where class $W[1]$ contains class FPT and can be regarded as a parameterized version of the classical complexity class NP. A parameterized problem that is $W[t]$-hard for any $W[t]$ in the hierarchy is unlikely to be FPT and is thus *fixed-parameter intractable*.

The basic tool for establishing fixed-parameter intractability is the following type of reductions between parameterized problems, which are formulated as decision problems. A *parametric reduction* from a parameterized problem Π to another parameterized problem Π' is an FPT algorithm that computes for each instance (I, k) of Π an instance (I', k') of Π' such that

(i) $k' \leq g(k)$ for some computable function $g(k)$, and
(ii) (I, k) is a 'Yes'-instance of Π iff (I', k') is a 'Yes'-instance of Π'.

A parametric reduction from Π to Π' ensures that $\Pi \in$ FPT whenever $\Pi' \in$ FPT, and Π' is $W[t]$-hard whenever Π is $W[t]$-hard. We refer the readers to the monograph of Downey and Fellows [2] and a recent book of Flum and Grohe [6] for a comprehensive treatment on parameterized complexity.

All graphs in this paper are undirected simple graphs and we assume that the readers are familiar with the basic terms in graph theory. We follow standard notation in graph theory (see the textbook of West [7], for instance) with the convention that m and n denote the numbers of edges and vertices, respectively, of the input graph $G = (V, E)$. We may also use $V(G)$ and $E(G)$ to denote the vertex and edge sets, respectively, of G, and we use $\bar{G}$ for the complement of G.

A vertex *covers* edges incident with it and *dominates* vertices adjacent to it. Similarly, an edge *covers* its two ends, and *dominates* edges adjacent to it. For a vertex v, $d_G(v)$ denotes the degree of v, and $N_G(v)$ the *neighbourhood* of v, i.e. the set of vertices adjacent to v. For a subset V' of vertices, $e_G(V')$ denotes the number of edges covered by vertices in V', $G[V']$ the subgraph induced by V'. The *open neighbourhood* $N_G(V')$ of V' is the set of vertices in $V - V'$ that are adjacent to vertices in V', and $N_G(V') \cup V'$ forms the *closed neighbourhood* $N_G[V']$ of V'. For a subset E' of edges, $V(E')$ denotes the set of vertices incident with edges in E', $G[E']$ the partial subgraph consisting of vertices $V(E')$ and edges E'. The *open edge-neighbourhood* $N_G(E')$ of E' is the set of edges in $E - E'$ that are adjacent to edges in E', and $N_G(E') \cup E'$ forms the *closed edge-neighbourhood* $N_G[E']$ of E'. We will drop subscripts in our notation when there is no confusion.

For any two disjoint sets X and Y of vertices in G, $[X, Y]$ denotes the edge set $\{xy : x \in X \text{ and } y \in Y\}$ and $e(X, Y)$ the number of edges in $[X, Y]$. When $Y = V - X$, $[X, Y]$ forms a *cut*. A set V' of vertices is a *clique* if $G[V']$ is a complete graph, an *independent set* if $G[V']$ has no edge, a *vertex cover* if V' covers all edges, and a *dominating set* if V' dominates all vertices in $V - V'$. A set E' of edges is a *matching* if edges in E' are mutually nonadjacent, and an *induced matching* if $G[V(E')]$ consists of a matching.

For a graph $G = (V, E)$, its *incidence graph* G' is the bipartite graph obtained from G by subdividing each edge e by a new vertex v_e. For vertices in G', each vertex of G is a *V-vertex* and each inserted vertex v_e is an *E-vertex*.

We also need the following notion of universal sets for the random separation method in Section 3.4. A list of binary vectors of length n is *(n, t)-universal* if for every subset of size t of the indices, all 2^t configurations appear in the list. Finally, we use $\tilde{O}(f(n))$ as a shorthand of $O(f(n)\log^c n)$, where c is a constant.

1.3. Problems and related work

We will consider around 20 graph problems and their parametric duals in this paper. Following the convention in complexity theory, we specify each problem as a decision problem. First, let us recall the definitions of parameterized versions of three classical NP-complete graph problems.

k-Clique
Instance: Graph $G = (V, E)$ and non-negative integer k.
Parameter: k.
Question: Does G contain a clique of size at least k?

Independent k-Set
Instance: Graph $G = (V, E)$ and non-negative integer k.
Parameter: k.
Question: Does G contain an independent set of size at least k?

k-Vertex Cover
Instance: Graph $G = (V, E)$ and non-negative integer k.
Parameter: k.
Question: Does G have a vertex cover of size at most k?

Both k-Clique and Independent k-Set are $W[1]$-complete as proved by Downey and Fellows [2, 8], but k-Vertex Cover is solvable in $O(kn + 1.2738^k)$ time by an algorithm of Chen *et al.* [9].

We now define various k-cardinality optimization problems in connection with some fundamental relations among vertices and edges. For simplicity, we will only define maximization versions of these problems. Their minimization and exact-value versions can be easily obtained by changing 'at least' in the *Question* part to 'at most' and 'exactly', respectively. Furthermore, the parametric dual of a problem Π can be obtained by replacing k in the *Question* part of Π by $n - k$ or $m - k$ depending on whether a solution for Π consists of k vertices or k edges.

We start with fixed-cardinality optimization problems dealing with edges incident with k vertices.

Maximum k-Vertex Cover (MaxVC(k))
Instance: Graph $G = (V, E)$, non-negative integers l and k.
Parameter: k.
Question: Does G have k vertices that cover at least l edges?

Maximum k-Vertex Subgraph (MaxVS(k))
Instance: Graph $G = (V, E)$, non-negative integers l and k.
Parameter: k.
Question: Does G have k vertices V' such that the induced subgraph $G[V']$ contains at least l edges?

Maximum $(k, n - k)$-Cut (MaxCut(k))
Instance: Graph $G = (V, E)$, non-negative integers l and k.
Parameter: k.
Question: Does G have k vertices V' such that the cut $[V', V - V']$ contains at least l edges?

Related work. MaxVS(k) and MinVS(k) [equivalently, MinVC($n - k$) and MaxVC($n - k$)] contain k-Clique and Independent k-Set, respectively, as special cases, and are thus $W[1]$-hard following the $W[1]$-completeness of k-Clique and Independent k-Set by Downey and Fellows [8]. Downey *et al.* [3] have independently established the $W[1]$-hardness of MaxCut(k), where they call the problem Cutting k Vertices From A Graph. MaxVC(k) and MaxVS(k) have been well studied in the literature in terms of approximation algorithms, where MaxVS(k) is also referred to as the *dense subgraph problem*. Asahiro *et al.* [10] have considered the complexity of MaxVS(k) when k is a part of input and the number l of edges is some function of k. MaxCut(k) is a fixed-cardinality maximization counterpart of NP-complete problems Max Cut ([ND16] in [11]) and Bipartite Subgraph ([GT25] in [11]), and approximation algorithms for MaxCut(k) have been studied in the literature. We refer the reader to an article of Feige and Langberg [12] that contains a mini survey on approximation algorithms for MaxVC(k), MaxVS(k), and MaxCut(k).

The next two problems are concerned with vertices adjacent to k vertices.

Maximum k-Vertex Domination (MaxVD(k))
Instance: Graph $G = (V, E)$, non-negative integers l and k.
Parameter: k.
Question: Are there k vertices V' in G whose open neighbourhood $N(V')$ has at least l vertices?

Maximum k-Vertex Joint (MaxVJ(k))
Instance: Graph $G = (V, E)$, non-negative integers l and k.
Parameter: k.
Question: Are there k vertices V' in G that contains at least l vertices adjacent to vertices in $V - V'$?

Related work. MaxVD(k) is equivalent to MaxVJ($n - k$) and contains the parameterized version Dominating k-Set of the classical NP-complete problem Dominating Set ([GT2] in [11]) as a special case. Therefore, both MaxVD(k) and MaxVJ($n - k$) are W[2]-hard following the W[2]-completeness of Dominating k-Set by Downey and Fellows [2]. MinVD(k) is equivalent to MinVJ($n - k$) and is related to the vertex connectivity of G in the sense that it is also equivalent to the problem of removing at most l vertices to cut off k vertices from G. In this connection, the W[1]-hardness of MinVD(k) has been independently established by Marx [4], where he calls the problem Separating k Vertices.

In the following four problems, we consider various ways edges are connected to vertices and edges.

Maximum k-Edge Cover (MaxEC(k))
Instance: Graph $G = (V, E)$, non-negative integers l and k.
Parameter: k.
Question: Are there k edges E' in G that cover at least l vertices, i.e. the partial subgraph $G[E']$ contains at least l vertices?

Maximum k-Edge Subgraph (MaxES(k))
Instance: Graph $G = (V, E)$, non-negative integers l and k.
Parameter: k.
Question: Does G have k edges E' such that the induced subgraph $G[V(E')]$ has at least l edges?

Maximum k-Edge Domination (MaxED(k))
Instance: Graph $G = (V, E)$, non-negative integer l and k.
Parameter: k.
Question: Does G have k edges E' that are adjacent to at least l edges in $E - E'$?

Maximum k-Edge Link (MaxEL(k))
Instance: Graph $G = (V, E)$, non-negative integer l and k.
Parameter: k.
Question: Are there k edges E' in G such that at least l edges connect $G[E']$ to the outside, i.e. the cut $[V(E'), V - V(E')]$ contains at least l edges?

Related work. Both MaxEC(k) and MaxED(k) can be regarded as fixed-cardinality maximization counterparts of the classical maximum matching problem, which is well known to be solvable in polynomial time. MaxED(k) is also a fixed-cardinality maximization version of Edge Dominating Set (see [GT2] in [11]), and k-Edge Domination on G is equivalent to k-Vertex Domination on the line graph of G.

In the last two problems, we deal with the number of components generated by removing k vertices or k edges.

Maximum k-Vertex Multicomponent Cut (MaxVMC(k))
Instance: Graph $G = (V, E)$, non-negative integer l and k.
Parameter: k.
Question: Does G have k vertices V' such that $G - V'$ has at least l components?

Maximum k-Edge Multicomponent Cut (MaxEMC(k))
Instance: Graph $G = (V, E)$, non-negative integer l and k.
Parameter: k.
Question: Does G have k edges E' such that $G - E'$ has at least l components?

Related work. MaxVMC(k) and MaxEMC(k) are fixed-cardinality counterparts of the well-studied multiway cut problems: remove the minimum number of edges (vertices) to cut a graph into k components. Recently, Marx [4] has independently proved the W[1]-hardness of MaxVMC(k) under the name of Separating Into l Components, and Downey *et al.* [3] have proved that the W[1]-hardness of the following *Graph k-Cut* problem that is closely related to MaxEMC(k): remove as few edges as possible to create at least k components.

1.4. Outline of the paper

For the fixed-cardinality optimization problems and their parametric duals defined in Section 1.3, we prove in Section 2 that around half of these problems are W[1]-hard. In contrast to the usual complication of parametric reductions, reductions in all our proofs are short and simple though subtle sometimes. In Section 3, we give polynomial-time and FPT algorithms for around 10 problems. In Section 4, we present faster exact algorithms for several W[1]-hard fixed-cardinality optimization problems concerning edges incident with k vertices. For this purpose, we first give an algorithm for finding a minimum (maximum) weight triangle in a weighted graph, and then transform our problems to problems of finding (minimum) maximum weight triangles in auxiliary graphs. We discuss further research directions in Section 5.

Our results on fixed-parameter tractability and intractability are summarized in Table 1, and our results on exact algorithms are summarized in Table 2.

2. FIXED-PARAMETER INTRACTABILITY

In this section, we establish the fixed-parameter intractability of around half of the problems defined in Section 1.3 by proving their W[1]-hardness. We make a few comments before getting into our proofs. First, because our main concern here is the fixed-parameter intractability, we will not deal with the W[1]-membership issue though most of our problems belong to W[1]. Second, it is easy to see that, under normal circumstance, a fixed-cardinality exact-value problem is at least as hard as its corresponding fixed-cardinality optimization problems. Therefore, the W[1]-hardness of a fixed-cardinality maximization (minimization) problem implies the

TABLE 1. Parameterized complexity of fixed-cardinality optimization problems defined in Section 1.3, where a '?' indicates an open problem.

Problem	Min	Max
VERTEX COVER(k), VERTEX SUBGRAPH($n-k$)	$W[1]$-hard (Section 2.2)	$W[1]$-hard (Section 2.2)
VERTEX COVER($n-k$), VERTEX SUBGRAPH(k)	$W[1]$-hard [8] (Section 2.2)	$W[1]$-hard [8] (Section 2.2)
CUT(k), CUT($n-k$)	$W[1]$-hard[a] (Section 2.3)	$W[1]$-hard (Section 2.3)
VERTEX DOMINATION(k), VERTEX JOINT($n-k$)	$W[1]$-hard[b] (Section 2.3)	W[2]-hard [8]
VERTEX DOMINATION($n-k$), VERTEX JOINT(k)	$W[1]$-hard (Section 2.3)	$O(m+n+4^k k^{3/2})$ (Section 3.3.1)
EDGE COVER(k)	$W[1]$-hard (Section 2.4)	$O(k(m+n))$ (Section 3.2.2)
EDGE COVER($m-k$)	FPT [5]	$O(m\sqrt{n})$ (Section 3.2.2)
EDGE SUBGRAPH(k)	?	$W[1]$-hard (Section 2.4)
EDGE SUBGRAPH($m-k$)	FPT [5]	$O(km+9.5^k k^2 n)$ (Section 3.3.3)
EDGE DOMINATION(k)	$W[1]$-hard (Section 2.4)	$W[1]$-hard (Section 2.4)
EDGE DOMINATION($m-k$)	$4^k\, k^{O(\log k)}\,(m+kn)\log n$ (Section 3.4.1)	$O(km+4^k k^{3/2} n)$ (Section 3.3.2)
EDGE LINK(k)	$W[1]$-hard (Section 2.4)	$W[1]$-hard (Section 2.4)
EDGE LINK($m-k$)	FPT [5]	FPT [5]
VERTEX MULTICOMPONENT CUT(k)	$O(m+n)$ (Section 3.2.1)	$W[1]$-hard[c] (Section 2.5)
VERTEX MULTICOMPONENT CUT($n-k$)	$O(m+n)$ (Section 3.2.1)	$W[1]$-hard (Section 2.5)
EDGE MULTICOMPONENT CUT(k)	$O(m+n)$ (Section 3.2.1)	?
EDGE MULTICOMPONENT CUT($m-k$)	$O(m+n)$ (Section 3.2.1)	$W[1]$-hard (Section 2.5)

[a]Proved independently by Downey *et al.* [3], where the problem is called CUTTING k VERTICES FROM A GRAPH.
[b]Proved independently by Marx [4], where the problem is called SEPARATING k VERTICES.
[c]Proved independently by Marx [4], where the problem is called SEPARATING INTO l COMPONENTS.

TABLE 2. Running time of faster exact algorithms for fixed-cardinality problems concerning edges incident with k vertices, where $\omega < 2.376$. The last column Xct refers to fixed-cardinality exact-value problems.

Problem	Min	Max	Xct
VC(k), VS($n-k$) (Section 4.2)	$\tilde{O}(kn^{\omega\lfloor k/3\rfloor+1+k \bmod 3})$	$\tilde{O}(kn^{\omega\lfloor k/3\rfloor+1+k \bmod 3})$	$O(k^2 n^{\omega\lfloor k/3\rfloor+2+k \bmod 3})$
VS(k), VC($n-k$) (Section 4.3)	$\tilde{O}(k^2 n^{\omega\lfloor k/3\rfloor+k \bmod 3})$	$\tilde{O}(k^2 n^{\omega\lfloor k/3\rfloor+k \bmod 3})$	$O(k^4 n^{\omega\lfloor k/3\rfloor+\text{ k mod 3}})$
CUT(k), CUT($n-k$) (Section 4.3)	$\tilde{O}(kn^{\omega\lfloor k/3\rfloor+1+k \bmod 3})$	$\tilde{O}(kn^{\omega\lfloor k/3\rfloor+1+\ k \bmod 3})$	$O(k^2 n^{\omega\lfloor k/3\rfloor+2+k \bmod 3})$

$W[1]$-hardness of its corresponding fixed-cardinality exact-value problem. Third, all our reductions clearly take FPT time and we will omit proofs for this fact. Fourth, when k is a part of input, all our parametric reductions become normal polynomial reductions and therefore all our $W[1]$-hard problems become NP-hard. Finally, we note that k-CLIQUE on regular graphs plays a central role as it is a special case of many of our $W[1]$-hard problems. We organize our proofs into Sections 2.1–2.5.

2.1. Cliques and induced matchings

To lay the foundation of our $W[1]$-hardness results, we first establish the fixed-parameter intractability on regular graphs for k-CLIQUE[2], INDEPENDENT k-SET and the following INDUCED k-MATCHING problem.

[2]Marx [4] has also observed recently and independently that k-CLIQUE is $W[1]$-hard on regular graphs.

INDUCED k-MATCHING
Instance: Graph $G = (V, E)$, and non-negative integer k.
Parameter: k.
Question: Does G have an induced matching $M \subseteq E$ with k edges, i.e. k edges M such that $G[V(M)]$ consists of k mutually non-adjacent edges?

We will see that almost all our parametric reductions are from these three problems. Note that the degree constraints on regular graphs are required for some of our parametric reductions from k-CLIQUE.

THEOREM 2.1. *For any fixed integers α, $c \geq 1$, k-*CLIQUE *is $W[1]$-complete for d-regular graphs with $d \geq \alpha k^c$.*

Proof. We give a parametric reduction from k-CLIQUE on general graphs. Let G be a graph and Δ its maximum degree. We construct a d-regular graph G' with $d \geq \alpha k^c$ as follows:

(i) Let $d = \max\{\Delta, \alpha\, k^c\}$.

(ii) Take d distinct copies $G_1, G_2, \ldots, G_d$ of G, and let v_i denote the vertex in G_i that corresponds to vertex v in G.
(iii) For every vertex v in G, create $d - d_G(v)$ distinct vertices V_v and for each vertex v_i, $1 \le i \le d$, add edges between v_i and every vertex in V_v.

Note that G' has at most $2dn$ vertices and d^2n edges, and can be constructed in $O(d^2n)$ time, which is actually polynomial time. We also note that the graph in G' formed by edges not in any G_i is a bipartite graph. Therefore, G contains a k-clique iff G' contains a k-clique, which establishes the theorem. □

Since Independent k-Set for G is equivalent to k-Clique for the complement $\bar{G}$ of G, we immediately have the following result.

Corollary 2.2. Independent k-Set *is* $W[1]$*-complete for regular graphs.*

Now we establish the fixed-parameter intractability of Induced k-Matching.

Theorem 2.3. Induced k-Matching *is* $W[1]$*-hard on regular graphs.*

Proof. We give a parametric reduction from Independent k-Set on regular graphs. Let $G = (V, E)$ be a regular graph. Construct a new regular graph H from G as follows:

(i) Take a copy $G' = (V', E')$ of G.
(ii) For each vertex v in G, connect it with its corresponding vertex v' in G' and every vertex in $N_{G'}(v')$.

Note that this construction automatically connects each vertex v' in G' with its corresponding vertex v in G and every vertex in $N_G(v)$. Therefore, H is a regular graph since G is a regular graph. We claim that G contains an independent k-set iff H has an induced k-matching, and, therefore, the theorem follows from Corollary 2.2.

Suppose that S is an independent k-set in G. Then the k edges in H connecting vertices in S and their corresponding vertices in G' form an induced k-matching.

Conversely, suppose that M is an induced k-matching in H. Let $M_G = M \cap E$, $M_{G'} = M \cap E'$ and M^* be the edges in M across G and G'. Let M' be the corresponding edges of $M_{G'}$ in G. Then M' shares no vertex with $M_G \cup M^*$. We claim that $M_G \cup M' \cup M^*$ is also an induced k-matching in H.

Clearly, no vertex of M' is adjacent to vertices of M^* since no vertex of $M_{G'}$ is adjacent to vertices of M^*. Suppose that some vertex u of M_G is adjacent to a vertex v of M'. Then by the construction of H, u is adjacent to the corresponding vertex v' of v in $M_{G'}$, contradicting $M_G \cup M_{G'}$ being an induced matching in H. It follow that $M_G \cup M' \cup M^*$ is an induced k-matching in H, and vertices of $M_G \cup M'$ contain an independent set S_G with $|M_G \cup M'|$ vertices. Let S^* be the vertices of M^* in G. Then $S = S_G \cup S^*$ consists of k vertices in G. Since $M_G \cup M' \cup M^*$ is an induced matching, vertices in S are mutually non-adjacent, and thus S is an independent k-set in G. □

2.2. Vertex covers and induced subgraphs

We consider the eight fixed-cardinality optimization problems in connection with k-Vertex Cover and k-Vertex Subgraph. As mentioned in Section 1.3, both MaxVS(k) [equivalently MinVC($n - k$)] and MinVS(k) [equivalently MaxVC($n - k$)] are $W[1]$-hard as k-Clique and Independent k-Set, respectively, are their special cases. Here, we will show that for regular graphs, these eight problems are basically equivalent and are all $W[1]$-hard.

Lemma 2.4. *Let G be a regular graph and V' a subset of k vertices in G. Then the following statements are equivalent.*

(i) *V' is a maximum k-vertex cover in G.*
(ii) *$G[V']$ is a minimum k-vertex subgraph in G.*
(iii) *V' is a minimum k-vertex cover in $\bar{G}$.*
(iv) *$\bar{G}[V']$ is a maximum k-vertex subgraph in $\bar{G}$.*

Proof. Let d be the vertex degree of G, and m' the number of edges in $G[V']$. Then $\sum_{v \in V'} d(v) = dk$, and V' covers

$$\sum_{v \in V'} d(v) - m' = dk - m'$$

edges. This clearly gives the equivalence between (i) and (ii), and between (iii) and (iv). Since the total number of edges in $G[V']$ and $\bar{G}[V']$ is $\binom{k}{2}$, (ii) and (iv) are equivalent. □

Since MaxVC(k) is equivalent to MinVS($n - k$), Lemma 2.4 indicates that for regular graphs, all eight fixed-cardinality optimization problems concerning k-vertex covers and k-vertex subgraphs are essentially the same problem. Combining Theorem 2.1 with Lemma 2.4, we obtain the following theorem.

Theorem 2.5. *The following eight problems are* $W[1]$*-hard on regular graphs.*

(i) Minimum k-Vertex Cover.
(ii) Maximum k-Vertex Cover.
(iii) Minimum $(n - k)$-Vertex Cover.
(iv) Maximum $(n - k)$-Vertex Cover.
(v) Minimum k-Vertex Subgraph.
(vi) Maximum k-Vertex Subgraph.
(vii) Minimum $(n - k)$-Vertex Subgraph.
(viii) Maximum $(n - k)$-Vertex Subgraph.

2.3. Cuts and domination

Here, we consider $(k, n - k)$-Cut, k-Vertex Domination and k-Vertex Joint problems and their parametric duals. As mentioned in Section 1.3, MaxVD(k) is equivalent to MaxVJ($n - k$)

and MinVD(k) is equivalent to MinVJ($n - k$). Furthermore, MinVD(k) and MinVJ($n - k$) are equivalent to the problem of deleting at most l vertices to cut off k vertices, which can be regarded as a vertex version of MinCut(k) where we wish to delete at most l edges to cut off k vertices.

Let us start with (k, $n - k$)-CUT. Clearly, the problem is its own parametric dual, and MinCut(k) for G is equivalent to MaxCut(k) for the complement $\bar{G}$ of G. It turns out that for regular graphs, k-CLIQUE and INDEPENDENT k-SET, respectively, are special cases of MinCut(k) and MaxCut(k).

THEOREM 2.6. *Both* MINIMUM (k, $n - k$)-CUT *and* MAXIMUM (k, $n - k$)-CUT *are* $W[1]$*-hard on regular graphs.*

Proof. For a d-regular graph G, it is clear that G has a k-clique iff G has a (k, $n - k$)-cut with $\leq kd - 2\binom{k}{2}$ edges, and G has an independent k-set iff G has a (k, $n - k$)-cut with $\geq kd$ edges. The theorem follows from the $W[1]$-hardness of k-CLIQUE and INDEPENDENT k-SET on regular graphs. □

We now establish the $W[1]$-hardness of MinVD(k) and MinVJ($n - k$).

THEOREM 2.7. MINIMUM k-VERTEX DOMINATION [*equivalently*, MINIMUM ($n - k$)-VERTEX JOINT) *is* $W[1]$*-hard.*

Proof. We give a parametric reduction from MINIMUM k-VERTEX COVER on regular graphs. For an arbitrary instance (G, l, k) of MinVC(k), we construct an instance (G', l, k) of MinVD(k) as follows.

(i) Construct the incidence graph of G by subdividing each edge e of G by a vertex v_e.
(ii) Add a complete graph K on $l + 2$ vertices.
(iii) Connect every vertex in K to every E-vertex v_e by an edge.

We claim that G has k vertices covering at most l edges iff G' has k vertices whose open neighbourhood has at most l vertices.

Suppose that G has k vertices V' covering at most l edges. Since each vertex in the open neighbourhood $N_{G'}(V')$ of V' is an E-vertex v_e for some edge e of G that is covered by vertices in V', we have $|N_{G'}(V')| \leq l$. Conversely, suppose that G' has k vertices V' with $|N_{G'}(V')| \leq$ l. Since in G', each E-vertex or vertex in K has degree more than l, V' consists of V-vertices only. Note that if an edge e in G is covered by some vertex in V', then in graph G', the corresponding E-vertex v_e of e is a vertex in $N_{G'}(V')$. Therefore, V' covers at most l edges in G. □

THEOREM 2.8. MINIMUM k-VERTEX JOINT [e*quivalently*, MINIMUM ($n - k$)-VERTEX DOMINATION) *is* $W[1]$*-hard.*

Proof. We give a parametric reduction from k-CLIQUE on d-regular graphs with $d \geq k^2$. Let G be a d-regular graph satisfying $d \geq k^2$. Construct the incidence graph G' from G by subdividing each edge e of G by a vertex v_e. Set $k' = k + \binom{k}{2}$, and $l = k$. We claim that G has a k-clique iff G' has k' vertices that contain at most l vertices connected to the outside.

Suppose that V' is a k-clique in G. Let $U = \{v_e : e$ is an edge of $G[V']\}$. Then $V' \cup U$ is a set of $k + \binom{k}{2} = k'$ vertices in G'. Note that the degree of each vertex of V' in G' is $d \geq k^2 \geq k'$. Therefore, in $V' \cup U$, every vertex in V' is connected to the outside, but no vertex in U is connected to the outside. It follows that $V' \cup U$ is a set of k' vertices in G' that contains at most $l = k$ vertices connected to the outside.

Conversely, suppose that V^* is a set of k' vertices in G' that has at most $l = k$ vertices connected to the outside. Let X be the set of V-vertices in V^* and Y the set of E-vertices in V^*. Since every vertex in X has degree at least k', every vertex in X is connected to some vertices outside V^*. It follows that $|X| \leq k$ and Y contains at most $k - |X|$ vertices connected to some vertices outside V^*. Note that every E-vertex is adjacent to exactly two V-vertices. Therefore, a vertex v_e in Y is connected to some vertices outside V^* iff at least one of its two adjacent V-vertices are not in X, i.e. e is not an edge of $G[X]$. Since Y has $k' - |X|$ vertices, it follows that Y has at least

$$k' - |X| - (k - |X|) = \binom{k}{2}$$

vertices v_e such that e is an edge of $G[X]$. Therefore, $G[X]$ has at least $\binom{k}{2}$ edges and thus X is a k-clique in G as it has at most k vertices. □

We note that, as mentioned in Section 1.3, MaxVD(k) and MaxVJ($n - k$) are $W[2]$-hard as they contain the $W[2]$-complete problem DOMINATING k-SET as a special case. On the other hand, MaxVD($n - k$) and its equivalent problem MaxVJ(k) are FPT as we will see in Section 3.3.1.

2.4. Edge induced subgraphs

Let E' be a set of k or $m - k$ edges. We consider problems in connection with the number of vertices in $G[E']$, the number of edges in $G[V(E')]$, the number of other edges adjacent to E', and the number of edges connecting $G[E']$ with vertices outside $G[E']$. Among the 16 problems related to the above four quantities defined by E', six are $W[1]$-hard, nine are FPT, but MINIMUM k-EDGE SUBGRAPH is open.

Let us start with the number of vertices in $G[E']$. We establish the $W[1]$-hardness of MINIMUM k-EDGE COVER by showing that k-CLIQUE is a special case of the problem. On the other hand, we will see in Section 3.2.2 that both MAXIMUM k-EDGE COVER and MAXIMUM ($m - k$)-EDGE COVER are FPT by the matching technique. We also note that Cai *et al.* [5] have obtained an FPT algorithm for MINIMUM ($m - k$)-EDGE COVER by using their newly developed random separation method.

Theorem 2.9. Minimum k-Edge Cover *is $W[1]$-hard on regular graphs.*

Proof. Let E' be a set of $\binom{k}{2}$ edges in a graph G, and l the number of vertices in $G[E']$. Then the maximum number of edges in $G[E']$ in terms of l equals $\binom{l}{2}$. Therefore,

$$\binom{l}{2} \geq \binom{k}{2},$$

which yields $l \geq k$. Furthermore, $l = k$ iff $G[E']$ is a complete graph. Therefore, G has a k-clique iff $G[E']$ contains at most k vertices, and the theorem follows from the $W[1]$-hardness of k-Clique on regular graphs. □

Next, we consider the number of edges in $G[V(E')]$. We prove that Maximum k-Edge Subgraph is $W[1]$-hard by a reduction from k-Clique. On the other hand, we will see in Section 3.3.3 that Maximum $(m-k)$-Edge Subgraph is FPT by kernelization. Furthermore, Minimum $(m-k)$-Edge Subgraph is recently shown to be FPT by Cai *et al.* [5] using their random separation method, but Minimum k-Edge Subgraph is open.

Theorem 2.10. Maximum k-Edge Subgraph *is $W[1]$-hard on regular graphs.*

Proof. Let G be a regular graph. We construct a graph H from G by taking a copy G' of G and adding all possible edges between vertices of G and vertices of G'. It is clear that H is also a regular graph, and that G contains a k-clique iff H contains a $2k$-clique. Furthermore, H contains a $2k$-clique iff it contains k edges E' such that $H[V(E')]$ contains at least $\binom{2k}{2}$ edges. Therefore, G contains a k-clique iff H contains k edges E' such that $H[V(E')]$ has at least $l = \binom{2k}{2}$ edges. The theorem follows from the $W[1]$-hardness of k-Clique on regular graphs. □

We now turn to the size of the open edge-neighbourhood of k or $m-k$ edges, and establish the $W[1]$-hardness of Maximum k-Edge Domination and Minimum k-Edge Domination by showing that they contain Induced k-Matching and k-Clique, respectively, as special cases. On the other hand, Maximum $(m-k)$-Edge Domination is FPT by kernelization (see Section 3.3.2), and Minimum $(m-k)$-Edge Domination is FPT by the random separation method (see Section 3.4.1).

Theorem 2.11. Maximum k-Edge Domination *is $W[1]$-hard on regular graphs.*

Proof. For a d-regular graph G, k edges are adjacent to at least $2(d-1)k$ other edges iff G has an induced k-matching. The theorem directly follows from Theorem 2.3, the $W[1]$-hardness of Induced k-Matching on regular graphs. □

Theorem 2.12. Minimum k-Edge Domination *is $W[1]$-hard on regular graphs.*

Proof. Let G be a d-regular graph satisfying $d \geq k^2$. We show that G has a k-clique iff G has $\binom{k}{2}$ edges E' such that $|N(E')| \leq dk - 2\binom{k}{2}$. The theorem then follows from the $W[1]$-hardness of k-Clique on such regular graphs.

Let V' be a k-clique of G. By taking edges in $G[V']$ as E', we see that E' are adjacent to at most $dk - 2\binom{k}{2}$ other edges. Conversely, the number l of edges in the open edge-neighbourhood $N(E')$ of E' satisfies $l = dn' - m' - \binom{k}{2}$, where n' is the number of vertices in $G[E']$ and m' the number of edges in $G[V(E')]$. Since $m' \leq \binom{n'}{2}$, we have

$$l \geq dn' - \binom{n'}{2} - \binom{k}{2}.$$

Note that $k \leq n' \leq k(k-1)$ as E' has $\binom{k}{2}$ edges. Therefore, by the assumption that $d \geq k^2$, we have $k \leq n' < d$.

Since

$$f(t) = dt - \binom{t}{2} = \frac{t(2d+1-t)}{2}$$

monotonically increases for $1 \leq t \leq d + 1/2, dn' - \binom{n'}{2}$ attains its minimum for $n' = k$. Therefore, $l \geq dk - 2\binom{k}{2}$ and $l = dk - 2\binom{k}{2}$ iff $n' = k$. By the assumption that $|N(E')| \leq dk - 2\binom{k}{2}$, we see that $G[E']$ has exactly k vertices and $\binom{k}{2}$ edges and thus $V(E')$ is a k-clique in G. □

For the problems concerning edges connecting $G[E']$ with outside vertices, it turns out that k-Clique and Induced k-Matching, respectively, are special cases of Minimum k-Edge Link and Maximum k-Edge Link. On the other hand, both Minimum $(m-k)$-Edge Link and Maximum $(m-k)$-Edge Link are FPT by the random separation method [5].

Theorem 2.13. Minimum k-Edge Link *is $W[1]$-hard on regular graphs.*

Proof. Let G be a d-regular graphs satisfying $d \geq 2k^2$. We prove that G has a k-clique iff it has $\binom{k}{2}$ edges E' such that at most $dk - 2\binom{k}{2}$ edges connect $G[E']$ with vertices not in $G[E']$. The theorem then follows from the $W[1]$-hardness of k-Clique on such regular graphs.

By taking all edges in a complete subgraph on k vertices, we have $\binom{k}{2}$ edges E' such that at most $dk - 2\binom{k}{2}$ edges connect $G[E']$ with vertices not in $G[E']$.

Conversely, consider the number l of edges connecting $G[E']$ with vertices not in $G[E']$. Let n' be the number of vertices in $G[E']$ and m' the number of edges in $G[V(E')]$. Then

$$l = dn' - 2m' \geq dn' - 2\binom{n'}{2}$$

as $m' \leq \binom{n'}{2}$.

Similar to the proof of Theorem 2.12, we see that $k \leq n' < d/2$, and $dn' - 2\binom{n'}{2}$ monotonically increases for $1 \leq t \leq (d+1)/2$. Therefore, $dn' - 2\binom{n'}{2}$ attains its minimum for $n' = k$, i.e. $l \geq dk - 2\binom{k}{2}$ and $l = dk - 2\binom{k}{2}$ iff $n' = k$. Therefore, $l \leq dk - 2\binom{k}{2}$ iff $G[E']$ has exactly k vertices and $\binom{k}{2}$ edges, i.e. G has a k-clique. □

THEOREM 2.14. MAXIMUM k-EDGE LINK *is* $W[1]$*-hard on regular graphs.*

Proof. It is easy to see that a d-regular graph G contains an induced k-matching iff it contains k edges E' such that there are $2(d-1)k$ edges connecting $G[E']$ with vertices outside $G[E']$. The theorem follows from the $W[1]$-hardness of INDUCED k-MATCHING on regular graphs. □

2.5. Multicomponent cuts

There are eight fixed-cardinality optimization problems concerning the number of components generated by removing k vertices or k edges, and we prove that the following three are $W[1]$-hard.

(i) MAXIMUM k-VERTEX MULTICOMPONENT CUT,
(ii) MAXIMUM $(n-k)$-VERTEX MULTICOMPONENT CUT and
(iii) MAXIMUM $(m-k)$-EDGE MULTICOMPONENT CUT.

Among the remaining five problems, MAXIMUM k-EDGE MULTICOMPONENT CUT is the only open problem and the other four can be easily solved in $O(m+n)$ time (Section 3.2.1).

We start with MaxVMC$(n-k)$, which is easily seen to contain INDEPENDENT k-SET as a special case.

THEOREM 2.15. MAXIMUM $(n-k)$-VERTEX MULTICOMPONENT CUT *is* $W[1]$*-hard on regular graphs.*

Proof. For every graph G, it is clear that G has an independent k-set iff it contains $n-k$ vertices whose removal results in k isolated vertices, which form k components. The theorem follows from the $W[1]$-hardness of INDEPENDENT k-SET on regular graphs. □

On the other hand, the $W[1]$-hardness of MaxVMC(k) is less obvious.

THEOREM 2.16. MAXIMUM k-VERTEX MULTICOMPONENT CUT *is* $W[1]$*-hard.*

Proof. We give a parametric reduction from k-CLIQUE. Let (G, k) be an instance of k-CLIQUE. Without loss of generality, we may assume that G is a connected graph with at least k vertices. Construct an instance (G', l, k) of MaxVMC(k) from G as follows.

(i) Subdivide each edge e of G by a vertex v_e.
(ii) Add a complete graph K on $k+1$ new vertices and for each vertex of K connect it to every vertex of G by an edge.
(iii) Set l to $\binom{k}{2}+1$.

We prove that G contains a k-clique iff G' has k vertices whose removal results in a graph with at least l components, which will establish the theorem.

Suppose that V' is a k-clique in G. Then in graph $G' - V'$, every E-vertex that corresponds to an edge of $G[V']$ is an isolated vertex. These $\binom{k}{2}$ isolated vertices together with the component containing K form l components in $G' - V'$.

Conversely, suppose that V' is a set of k vertices in G' such that $G' - V'$ contains at least l components. Let V^* be V-vertices in V'. Note that the complete graph K contains at least one vertex of $G' - V'$, which is connected to all V-vertices not in V^*. Therefore, the deletion of vertices in $V' - V^*$, which are either E-vertices or vertices in K, from $G - V^*$ does not increase the number of components, and hence $G - V^*$ and $G - V'$ have the same number of components. In graph $G - V^*$, one component contains the complete graph K and each of the remaining $l-1$ components is formed by an isolated E-vertex v_e that corresponds to edge e in $G[V^*]$. Since $G' - V^*$ has at least $l - 1 = \binom{k}{2}$ such isolated vertices, $G[V^*]$ contains at least $\binom{k}{2}$ edges, which implies that V^* contains at least k vertices. Therefore, V^* has exactly k vertices and $G[V^*]$ contains exactly $\binom{k}{2}$ edges, implying that V^* is a k-clique in G. □

As the final $W[1]$-hardness result in this paper, we prove that MaxEMC$(m-k)$ is $W[1]$-hard by showing that MaxEMC$\left(m - \binom{k}{2}\right)$ is actually equivalent to k-CLIQUE.

THEOREM 2.17. MAXIMUM $(m-k)$-EDGE MULTICOMPONENT CUT *is* $W[1]$*-hard on regular graphs.*

Proof. We claim that for every graph G, G has a k-clique iff G contains $m - \binom{k}{2}$ edges E' whose deletion from G results in at least $n-k+1$ components. The theorem then follows from the $W[1]$-hardness of k-CLIQUE on regular graphs.

Let K be a k-clique in G and let E' be the edges of G not in $G[K]$. Then E' has $m - \binom{k}{2}$ edges and $G - E'$ consists of $G[K]$ and $n-k$ isolated vertices. Therefore, $G - E'$ contains $n-k+1$ components.

Conversely, suppose that G contains no k-clique. Let E' be an arbitrary set of $m - \binom{k}{2}$ edges in G and $G' = G - E'$. If G' has only one non-trivial component, then the component has at least $k+1$ vertices as it has $\binom{k}{2}$ edges but is not a k-clique. Therefore, G' has at most $n-(k+1)$ isolated vertices and thus at most $n-k$ components. Otherwise, let $H_1, \ldots, H_t$, $t \geq 2$, be non-trivial components of G'. Construct a new graph G^* from G' as follows: arbitrarily choose one vertex v_i from each H_i and, for each v_i with $i \geq 2$ replace every edge $v_i u$ by a new edge $v_1 u$. Therefore, G^* consists of isolated vertices v_i, $2 \leq i \leq t$, and a single non-trivial component H^*, which has $\binom{k}{2}$ edges and is not a k-clique. Furthermore, G' and G^* have the same number of components. By the previous argument, we see that G^* has at most $n - k$ components and hence G' has at most $n - k$ components. □

3. FPT ALGORITHMS

In this section, we consider fixed-parameter tractability of cardinality constrained optimization problems. We start with a general result in Section 3.1 that basically says we can focus on connected graphs for most fixed-cardinality optimization problems. Then we give polynomial-time algorithms or FPT algorithms for around 10 fixed-cardinality optimization problems listed in Table 1.

In Section 3.2, we give simple polynomial-time algorithms for minimum vertex/edge multicomponent cut problems and maximum edge cover problems. We use kernelization in Section 3.3 to solve Maximum $(n - k)$-Vertex Domination, Maximum $(m - k)$-Edge Domination and Maximum $(m - k)$-Edge Subgraph, and we use the random separation method in Section 3.4 to solve Minimum $(m - k)$-Edge Domination.

3.1. Disconnected graphs

To solve a problem on a disconnected graph G, we can usually solve the problem for each component of G individually and independently. Unfortunately, it is not the case for cardinality constrained optimization problems, and it is not obvious that an algorithm for a problem on connected graphs can be used directly to solve the problem for disconnected graphs. For instance, consider the problem of finding k induced copies of $K_1 \cup K_2$ to cover as many edges as possible. Nevertheless, for most cardinality constrained optimization problems, optimal solutions of G can be obtained from those of its components. For such problems, we will show that, as far as the fixed-parameter tractability is concerned, we need only consider connected graphs.

Definition 3.1. *A cardinality constrained optimization problem* Π *is* additive for components *if for any two vertex-disjoint graphs* G_1 *and* G_2, *the following two conditions hold*:

(i) *For every feasible* k_1*-solution* S_1 *of* G_1 *and every feasible* k_2*-solution* S_2 *of* G_2, $S_1 \cup S_2$ *is a feasible* $(k_1 + k_2)$*-solution of* $G_1 \cup G_2$ *and* $\phi(S_1 \cup S_2) = \phi(S_1) + \phi(S_2)$.
(ii) *Every optimal* k*-solution of* $G_1 \cup G_2$ *is the union of an optimal* k'*-solution of* G_1 *and an optimal* $(k - k')$*-solution of* G_2 for some $0 \leq k' \leq k$.

Most graph problems are additive for components, and we now prove that for any graph problem Π that is additive for components, Π is FPT for disconnected graphs whenever it is FPT for connected graphs.

Theorem 3.2. *Let* Π *be a* k*-cardinality optimization problem that is additive for components, and* $T(m, n, k)$ *the time to find an optimal* k*-solution and its value in a connected graph with* m *edges and* n *vertices. Let* $\alpha \geq 1$ *be a constant,* $f(k)$ *a computable function satisfying the property that* $f(1) > 0$ *and there is a constant* $c > 1$ *such that for all* $k \geq 2$, $f(k) \geq cf(k - 1)$. *Then the time to find an optimal* k*-solution of* Π *in a disconnected graph is*

(i) $O(f(k)(m + n)^\alpha)$ for $T(m, n, k) = O(f(k)(m + n)^\alpha)$,
(ii) $O(k(m + n)^\alpha + f(k)n)$ for $T(m, n, k) = O((m + n)^\alpha + f(k))$ and
(iii) $O(knT(m, n, k) + k^2 n)$ for any monotonically non-decreasing function $T(m, n, k)$ of m, n and k.[3]

Proof. We will only give a proof for maximization problems, which is easily adapted for minimization problems. Let Π be a k-cardinality maximization problem that is additive for components, $\mathcal{A}$ an algorithm for finding an optimal k-solution and its value for Π on connected graphs and G a disconnected graph with components $H_1, \ldots, H_t$. To find an optimal k-solution of G, we first use $\mathcal{A}$ to compute, for every component H_i, an optimal j-solution $S_j(H_i)$ and its value $\phi(S_j(H_i))$ for each j, $1 \leq j \leq k$. Once we have obtained $S_j(H_i)$ and $\phi(S_j(H_i))$ for all components and all values of j, we use dynamic programming to compute an optimal k-solution of G. For this purpose, we use two tables $S(i, j)$ and $F(i, j)$, $1 \leq i \leq t$, $0 \leq j \leq k$, where $S(i, j)$ contains an optimal j-solution in graph $H_1 \cup \cdots \cup H_i$ and $F(i, j)$ stores the value $\phi(S(i, j))$ of $S(i, j)$. Note that $S(t, k)$ and $F(t, k)$, respectively, hold an optimal k-solution of G and its value. We use the following recurrences to compute the entries of the two tables $S(i, j)$ and $F(i, j)$ row-by-row.
For each i, $1 \leq i \leq t$, $S(i, 0) = \varnothing$ and $F(i, 0) = 0$.
For each j, $1 \leq j \leq k$, $S(1, j) = S_j(H_1)$ and $F(1, j) = \phi(S(1, j))$.
For each i, $2 \leq i \leq t$ and each j, $1 \leq j \leq k$,

$$F(i,j) = \max_{0 \leq j' \leq j} \{F(i-1, j') + \phi(S_{j-j'}(H_i)\},$$

and

$$S(i,j) = S(i-1, j^*) \cup S_{j-j^*}(H_i),$$

where j^* is a value of j', $0 \leq j' \leq j$, that maximizes $F(i - 1, j') + \phi(S_{j-j'}(H_i))$.

In summary, we solve Π for disconnected graphs by the following algorithm.

[3]For most common functions $T(m, n, k)$, the time is actually $O(kT(m, n, k) + k^2 n)$.

Step 1. For each component H_i, $1 \leq i \leq t$, and each j, $1 \leq j \leq k$, find an optimal j-solution $S_j(H_i)$ of H_i and compute its value $\phi(S_j(H_i))$.
Step 2. Use the above recurrences to fill the entries of the two tables $S(i, j)$ and $F(i, j)$ row-by-row by dynamic programming.

We prove that the above algorithm correctly finds an optimal k-solution of G. For this purpose, we use induction on i to prove that for every $0 \leq j \leq k$, $S(i, j)$ is an optimal j-solution of H_i', where H_i' denotes the graph $H_1 \cup \cdots \cup H_i$, and $F(i, j)$ the value of $S(i, j)$. Let S be an optimal j-solution of H_i'. It suffices to show that $S(i, j)$ is a feasible j-solution of H_i' and $F(i, j) = \phi(S(i, j)) \geq \phi(S)$. By the recurrences for computing $S(i, j)$, we have

$$S(i,j) = S(i-1, j^*) \cup S_{j-j^*}(H_i)$$

for j^* being a value of j', $0 \leq j' \leq j$, that maximizes

$$F(i-1, j') + \phi(S_{j-j'}(H_i)).$$

Since $S_{j-j^*}(H_i)$ is an optimal $(j - j^*)$-solution of H_i and $S(i - 1, j^*)$ is an optimal j^*-solution of H'_{i-1} by the induction hypothesis, it follows from the first condition of Definition 3.1 that $S(i, j)$ is a feasible j-solution of H_i'.
Now by the second condition of Definition 3.1, S must be the union of an optimal j'-solution S_1 of H'_{i-1} for some $0 \leq j' \leq j$ and an optimal $(j - j')$-solution S_2 of H_i. By the induction hypothesis, $F(i - 1, j') = \phi(S(i - 1, j')) = \phi(S_1)$; and by the first condition of Definition 3.1, we have $\phi(S) = \phi(S_1) + \phi(S_2)$. It follows that

$$\begin{aligned}\phi(S) &= F(i-1, j') + \phi(S_{j-j'}(H_i)) \\ &\leq F(i-1, j^*) + \phi(S_{j-j^*}(H_i)) = \phi(S(i,j))\end{aligned}$$

and $F(i, j) = \phi(S(i, j))$, implying that indeed $S(i, j)$ is an optimal j-solution of H_i' and $F(i, j)$ contains the value of $S(i, j)$. This establishes the correctness of our algorithm.

We now consider the running time of our algorithm.

(i) Since $\sum_{i=1}^{t} (m_i + n_i) = m + n$ and $\alpha \geq 1$, we have

$$\sum_{i=1}^{t} (m_i + n_i)^\alpha \leq (m+n)^\alpha.$$

Also, we have $\sum_{j=1}^{k} f(j) < f(k) \sum_{j=0}^{\infty} c^{-j} = c/(\text{c-1}) f(k)$.

Therefore, the time taken by Step 1 is

$$\begin{aligned}\sum_{i=1}^{t}\sum_{j=1}^{k} T(m_i, n_i, j) &= O\left(\sum_{i=1}^{t}\sum_{j=1}^{k} f(j)(m_i+n_i)^\alpha\right) \\ &= O\left(\sum_{j=1}^{k} f(j) \sum_{i=1}^{t} (m_i+n_i)^\alpha\right) \\ &= O(f(k)(m+n)^\alpha).\end{aligned}$$

Step 2 takes $O(k^2 t) = O(k^2 n)$ time. Since $f(k) \geq c^k$ and $\alpha \geq 1$, we see that $k^2 n = O(f(k)(m+n)^\alpha)$. Therefore, the whole algorithm takes $O(f(k)(m+n)^\alpha)$ time.

(ii) The time taken by Step 1 is

$$\begin{aligned}\sum_{i=1}^{t}\sum_{j=1}^{k} T(m_i, n_i, j) &= O\left(\sum_{i=1}^{t}\sum_{j=1}^{k} \left[(m_i+n_i)^\alpha + f(j)\right]\right) \\ &= O\left(\sum_{j=1}^{k}\sum_{i=1}^{t} (m_i+n_i)^\alpha \right.\\ &\qquad \left. + \sum_{i=1}^{t}\sum_{j=1}^{k} f(j)\right) \\ &= O(k(m+n)^\alpha + f(k)n).\end{aligned}$$

Time for Step 2 is $O(k^2 n)$, which is dominated by the time for Step 1, and thus the whole algorithm takes $O(k(m+n)^\alpha + f(k)n)$ time.

(iii) The time taken by Step 1 is

$$\begin{aligned}\sum_{i=1}^{t}\sum_{j=1}^{k} T(m_i, n_i, j) &\leq \sum_{i=1}^{t} kT(m_i, n_i, k) \\ &\leq O(knT(m, n, k))\end{aligned}$$

as $T(m, n, k)$ is a monotonically non-decreasing function of m, n and k, and therefore the whole algorithm takes $O(knT(m, n, k) + k^2 n)$ time. □

For the parametric dual Π' of a k-cardinality constrained optimization problem Π, it is not clear in general whether the 'additive for components' property of Π is sufficient for Π' to be FPT for disconnected graphs when Π' is FPT for connected graphs. However, if for any two vertex-disjoint graphs G_1 and G_2, the size of the solution base of $G_1 \cup G_2$ equals the sum of the sizes of the solution bases of G_1 and G_2, then we obtain the same results as those in Theorem 3.2. Recall that for a k-cardinality optimization problem Π on G, $\bar{k}$ denotes $|\mathcal{B}(G)| - k$.

Theorem 3.3. *Let Π be a k-cardinality optimization problem that is additive for components, and T(m, n, k) the* *time to find an optimal $\bar{k}$-solution and its value for Π in a connected graph with m edges and n vertices. Let $\alpha \geq 1$ be a constant, $f(k)$ a computable function satisfying the property that $f(1) > 0$ and there is a constant $c > 1$ such that for all $k \geq 2$, $f(k) \geq cf(k-1)$. If for any two vertex-disjoint graphs G_1 and G_2, $|\mathcal{B}(G_1 \cup G_2)| = |\mathcal{B}(G_1)| + |\mathcal{B}(G_2)|$, then the time to find an optimal $\bar{k}$-solution of Π in a disconnected graph is*

(i) $O(f(k)(m+n)^{\alpha})$ *for* $T(m, n, k) = O(f(k)(m+n)^{\alpha})$,
(ii) $O(k(m+n)^{\alpha} + f(k)n)$ *for* $T(m, n, k) = O((m+n)\alpha + f(k))$ *and*
(iii) $O(knT(m, n, k) + k^2 n)$ *for any monotonically non-decreasing function* $T(m, n, k)$ *of m, n and k.*

Proof. Let $\mathcal{A}'$ be an algorithm that runs in $T(m, n, k)$ time and finds an optimal $\bar{k}$-solution and its value for Π on a connected graph G with m edges and n vertices. We use the the algorithm in the proof of Theorem 3.2 with the following changes to find an optimal $\bar{k}$-solution of G:

(i) $S_j(H_i)$ is an optimal $\bar{j}$-solution, instead of an optimal j-solution, of H_i.
(ii) $S(i, j)$ contains an optimal $\bar{j}$-solution of $H_1 \cup \cdots \cup H_i$.

By the assumption, we have $|\mathcal{B}(G_1 \cup G_2)| = |\mathcal{B}(G_1)| + |\mathcal{B}(G_2)|$ for any two vertex-disjoint graphs G_1 and G_2. The key point to note is that this ensures the following property: the union of a feasible $\overline{k'}$-solution of G_1 and a feasible $\overline{k-k'}$-solution of G_2 is a feasible $\bar{k}$-solution of $G_1 \cup G_2$ as

$$(|\mathcal{B}(G_1)| - k') + (|\mathcal{B}(G_2)| - (k-k')) = |\mathcal{B}(G_1)| + |\mathcal{B}(G_2)| - k = |\mathcal{B}(G_1 \cup G_2)| - k.$$

With this observation, we can prove the correctness of the algorithm using the same argument in the proof of Theorem 3.2.

For the running time of the algorithm, the proof is exactly the same as that for Theorem 3.2 except that the time $T(m, n, k)$ refers to the time of finding an optimal $\bar{k}$-solution. □

3.2. Polynomial-time algorithms

In this section, we give simple polynomial-time algorithms for Minimum Vertex Multicomponent Cut problems, Minimum Edge Multicomponent Cut problems, and Maximum Edge Cover problems.

3.2.1. Multicomponent cuts

First we consider the problem of deleting t vertices in a graph, where t is a part of input, to minimize the number of components in the resulting graph. Without loss of generality, we may assume that $m, n \geq t$. The problem can be easily solved in $O(m+n)$ time by the following greedy algorithm, which never increases the number of components in the graph.

Sort the components $H_1, \ldots, H_s$ of G in non-decreasing order with respect to their numbers of vertices. Let n_i be the number of vertices in H_i. Let j be the largest index such that $\sum_{i=1}^{j} n_i \leq t$. Delete all vertices in $H_1, \ldots, H_j$. For component H_{j+1}, construct a spanning tree T, repeatedly delete leaves of T until we have deleted exactly t vertices.

For the problem of deleting t edges in a graph, where t is a part of input, to minimize the number of components in the resulting graph, we have the following algorithm. For each component H_i, construct a spanning tree T_i and let E_i be non-tree edges in H_i. Let $E^* = \cup_{i=1}^{s} E_i$. If E^* has at least t edges, arbitrarily choose t edges from E^* and delete them. Otherwise, delete all edge in E^* and arbitrarily delete $t - |E^*|$ tree edges.

The above algorithm clearly runs in $O(m+n)$ time. It is also easy to verify the correctness of the algorithm, though not as obvious as the greedy algorithm for deleting t vertices, by noticing the following three facts.

(i) Edge deletion cannot reduce the number of components in a graph.
(ii) If G has s components, then $n - s$ is the minimum number of edges required to maintain s components. In such a case, each component is a tree.
(iii) For a tree, the deletion of any edge breaks up the tree into two components.

Theorem 3.4. *For every integer t, it takes $O(m+n)$ time to delete t vertices (or t edges) in a graph to minimize the number of components in the resulting graph.*

The above theorem clearly implies the linear-time solvability of the four fixed-cardinality minimization problems concerning the number of components by vertex or edge deletion.

Corollary 3.5. *It takes $O(m+n)$ time to solve* Minimum k-Vertex Multicomponent Cut, Minimum $(n-k)$-Vertex Multicomponent Cut, Minimum k-Edge Multicomponent Cut *and* Minimum $(m-k)$-Edge Multicomponent Cut.

3.2.2. Maximum edge covers

We consider the two problems MaxEC(k) and MaxEC($m-k$) that ask us to find k and $m-k$ edges, respectively, to cover the maximum number of vertices. We will use the matching technique to obtain an efficient polynomial-time algorithm for solving both problems.

Lemma 3.6. *Let G be a graph with at least k edges and no isolated vertices. Let M be a maximum matching of G. Then the maximum number of vertices that k edges can cover equals 2k if $k \leq |M|$, $k + |M|$ if $k \leq n - |M|$, and n otherwise.*

Proof. Obviously, k edges cover at most $2k$ vertices. If $k \leq |M|$, then any k edges from M cover $2k$ vertices. Otherwise, let E' be a set of k edges, and M' a maximum matching in $G[E']$.

Then, the number of vertices covered by E' is at most

$$2|M'| + (k - |M'|) = k + |M'| \leq k + |M|.$$

Note that there are $n - 2|M|$ M-unsaturated vertices in G. If $k \leq n - |M|$, then $k - |M| \leq n - 2|M|$ and thus $|M|$ edges in M together with $k - |M|$ edges incident with $k - |M|$ distinct M-unsaturated vertices cover exactly $k + |M|$ vertices. Otherwise $k > n - |M|$ and thus there are less than $k - |M|$ M-unsaturated vertices in G. Edges in M together with any $k - |M|$ edges that are incident with all M-unsaturated vertices cover all vertices in the graph. □

Based on the proof of the above lemma, we have the following efficient algorithm to find k edges E', where k is a part of input, to cover the maximum number of vertices in a graph G.

(i) Delete all isolated vertices from G.
(ii) If G has a matching M of size at least k, then arbitrarily take k edges from M as E' and stop.
(iii) Find a maximum matching M of G and put all edge in M into E'.
(iv) For each distinct M-unsaturated vertex v, arbitrarily put an edge incident with v into E' until either E' has k edges or there is no more M-unsaturated vertices.
(v) Arbitrarily, add $k - |E'|$ edges into E'.

Note that Steps (ii) and (iii) take $O(k(m + n))$ and $O(m\sqrt{n})$ time, respectively, by using the maximum matching algorithm of Micali and Vazirani [13]. Since all other steps takes linear time, we have the following theorem.

THEOREM 3.7. MAXIMUM k-EDGE COVER *can be solved in* $O(k(m + n))$ *time and* MAXIMUM $(m - k)$-EDGE COVER *can be solved in* $O(m\sqrt{n})$ *time.*

3.3. FPT by kernelization

The main idea of the kernelization method is to reduce a problem instance I in polynomial time to an "equivalent" instance I' whose size is bounded by a function of the parameter k, and then solve I' by exhaustive search or other methods. Here we will use the kernelization method to solve MAXIMUM $(n - k)$-VERTEX DOMINATION, MAXIMUM $(m - k)$-EDGE DOMINATION, and MAXIMUM $(m - k)$-EDGE SUBGRAPH.

3.3.1. Maximum (n − k)-vertex domination

We start with MAXIMUM $(n - k)$-VERTEX DOMINATION, the problem of finding $n - k$ vertices V' that maximize the number of vertices in the open neighbourhood $N(V')$ of V'. The problem is equivalent to MAXIMUM k-VERTEX JOINT, the problem of finding k vertices V^* to maximize the number of vertices in V^* adjacent to vertices outside V^*.

Recall that a set of vertices V' is a dominating set if $V - V' \subseteq N(V')$. The following upper bound on the number of vertices in a dominating set will enable us to reduce our problem to a kernel of size $\leq 2k$.

LEMMA 3.8 (Ore [14]). *Every graph G without isolated vertices has a dominating set of size at most* $\lfloor n/2 \rfloor$.

By the above lemma, we see that if $k < n/2$, then we can always find a set of $n - k$ vertices V' in G that form a dominating set, i.e. $N(V')$ contains all remaining k vertices. Otherwise G contains at most $2k$ vertices, which is regarded as a kernel, and we can use exhaustive search to find a set of $n - k \leq k$ vertices V' in G to maximize $|N(V')|$.

We can easily find a dominating set of size $\lfloor n/2 \rfloor$ in $O(m+n)$ time as follows: construct a spanning tree T of G and then take the smaller one of the two colour classes in the 2-colouring of T as the dominating set. For $n \leq 2k$, we note that $\binom{n}{k} \leq \binom{2k}{2} = O(4^k/\sqrt{k})$ and the time for determining the size of the open neighbourhood of k vertices is $O(k^2)$. Therefore, the time for exhaustive search is $O\big(\binom{2k}{2}k^2\big) = O(4^k k^{3/2})$, and the total time for the algorithm is $O(m + n + 4^k k^{3/2})$.

THEOREM 3.9. MAXIMUM $(n - k)$-VERTEX DOMINATION (*equivalently* MAXIMUM k-VERTEX JOINT) *can be solved in* $O(m + n + 4^k\, k^{3/2})$ *time.*

3.3.2. Maximum (m − k)-edge domination

To obtain an FPT algorithm for MAXIMUM $(m - k)$-EDGE DOMINATION, the problem of finding $m - k$ edges E' that are adjacent to the maximum number of edges in $E - E'$, we first establish a relation with a maximal matching in a connected graph and use the relation to reduce the problem for a connected graph to a kernel of size $\leq 2k$.

LEMMA 3.10. *Let G be a connected graph, and M a maximal matching of G. If* $|M| > m - k$, *then G has* $\leq 2k$ *vertices and* $\leq 2k - 1$ *edges.*

Proof. Since G is connected, we have $m \geq n - 1$. By the assumption that $|M| \geq m - k + 1$ and the fact that $|M| \leq \lfloor n/2 \rfloor$, we have

$$\left\lfloor \frac{n}{2} \right\rfloor \geq m - k + 1 \geq n - k,$$

which yields $n \leq 2k$. It follows that

$$m \leq |M| + k - 1 \leq \lfloor n/2 \rfloor + k - 1 \leq 2k - 1. \qquad \square$$

Based on the above lemma, we have the following algorithm for finding a set E' of $m - k$ edges in a connected graph G to maximize the number of edges in the open edge-neighbourhood $N(E')$ of E'.

Find a maximal matching M in G. If $m - k \geq |M|$ then take all edges in M and arbitrarily select $m - k - |M|$ edges in $E - M$ as E'. Otherwise G has $\leq 2k$ vertices and $\leq 2k - 1$ edges, which is regarded as a kernel, and we use exhaustive search to find a set of required $m - k$ edges in G.

The correctness of the algorithm follows from the simple fact that edges in every maximal matching are adjacent to all edges not in the matching. Since a maximal matching in a graph can be found in $O(m + n)$ time, the algorithm takes $O(m + n)$ time when $m - k \geq |M|$. For the exhaustive search part, by Lemma 3.10, G has at most $2k$ vertices and $2k - 1$ edges and thus $m - k < k$. Note that it takes $O(k^2)$ time to determine the number of edges in the open edge-neighbourhood of k edges. Therefore, the exhaustive search takes $O\left(\binom{2k}{2}k^2\right) = O(4^k k^{3/2})$ time, and the total time for the algorithm is $O(m + n + 4^k k^{3/2})$.

Theorem 3.11. *It takes $O(m + n + 4^k k^{3/2})$ time to solve* Maximum $(m - k)$-Edge Domination *on connected graphs.*

To solve Maximum $(m - k)$-Edge Domination for general graphs, we use the results in Section 3.1 for disconnected graphs. It is clear that the problem is additive for components and satisfies the conditions of Theorem 3.3. Since $f(k) = 4^k k^{3/2}$ satisfies $f(k) > 4f(k - 1)$, it follows from Theorem 3.3 that the algorithm takes $O(k(m + n) + 4^k k^{3/2} n) = O(km + 4^k k^{3/2} n)$ time for disconnected graphs.

Corollary 3.12. *It takes $O(km + 4^k k^{3/2} n)$ time to solve* Maximum $(m - k)$-Edge Domination.

3.3.3. *Maximum $(m - k)$-edge subgraph*

We now consider the problem of finding $m - k$ edges whose ends induce a subgraph with the maximum number of edges. First we consider the problem for a connected graph G. Let E^* be a set of k edges in G. An edge $e \in E^*$ is an *inside edge* if both ends of e are incident with edges in $E - E^*$. We call E^* a *maximum k-set* if it has the maximum number of inside edges among all k-sets of edges. For $m - k$ edges $E - E^*$, it is clear that $G[V(E - E^*)]$ contains the maximum number of edges iff E^* is a maximum k-set, and MaxES$(m - k)$ is equivalent to the problem of finding a maximum k-set E^* of G. Furthermore, the maximum number of inside edges in E^* is k iff G contains a spanning subgraph with $m - k$ edges. This connection with spanning subgraphs enables us to use the following result to reduce our problem to a kernel with at most $4k$ vertices and $4k - 1$ edges.

Lemma 3.13. *Let G be a connected graph, and G' the graph obtained from G by deleting all leaves. If every spanning subgraph of G has more than $m - k$ edges, then G' has at most $2k$ vertices and $2k - 1$ edges.*

Proof. Let m' and n' be the numbers of edges and vertices, respectively, of G'. Construct a graph G^* from G' by adding a new vertex and connecting it with every odd-degree vertex of G'. Then G^* contains an Eulerian circuit $C = e_1, e_2, \ldots, e_{m'+t}, e_1$, where t is the number of odd-degree vertices in G'. Let $C' = E(G') \cap \{e_1, e_3, \ldots, e_{2i+1}, \ldots\}$, and L the set of leaf-edges of G. Then $G[C' \cup L]$ is a spanning subgraph of G with

$$\left\lceil \frac{m' + t}{2} \right\rceil - \frac{t}{2} + (m - m') = m - \left\lfloor \frac{m'}{2} \right\rfloor$$

edges. By the assumption that $m - \lfloor m'/2 \rfloor > m - k$, we deduce that $m' \leq 2k - 1$ and thus $n' \leq m' + 1 \leq 2k$ as G' is connected. □

The proof of the above theorem implies that if the leaf-deleted graph G' of G has more than $2k$ vertices or $2k - 1$ edges, then we can find in $O(m + n)$ time a spanning subgraph of G with $m - k$ edges, which yields a solution to MaxES$(m - k)$. Therefore, we need only consider the case that G' has at most $2k$ vertices and $2k - 1$ edges.

Construct a graph H from G as follows: for every non-leaf vertex v of G, delete all but one leaf adjacent to v. Let L be the set of deleted leaf-edges in forming H. Since G' has at most $2k$ vertices and $2k - 1$ edges, H contains at most $4k$ vertices and $4k - 1$ edges, and is taken as a kernel for our problem. Note that a maximum k-set E^* of G consists of k' edges inside G' for some $0 \leq k' \leq k$ and $k - k'$ leaf-edges of G.

Note that no leaf-edge of G can be an inside edge. Furthermore, if $|L| \geq k - k'$ then E^* can be obtained from a maximum k'-set of G' and arbitrary $k - k'$ edges from L; otherwise, E^* can be obtained by taking L, some k' edges inside G', and some $k - k' - |L|$ leaf-edges of H. In the former case, we consider all k'-subsets of edges in G', which takes at most $O(2^{2k} k^2)$ time. In the latter case, we consider all possible $(k - |L|)$-subsets of edges in H, which takes at most $\binom{4k}{2} k^2 = O(9.5^k k^2)$ time. Therefore, our algorithm takes $O(m + n + 9.5^k k^2)$ time.

Theorem 3.14. *It takes $O(m + n + 9.5^k k^2)$ time to solve* Maximum $(m - k)$-Edge Subgraph *for connected graphs.*

To solve our problem for general graphs, we use the results in Section 3.1 for disconnected graphs. Since $f(k) = 9.5^k\, k^2$ satisfies $f(k) > 9.5f(k - 1)$, the following result follows from Theorem 3.3.

Corollary 3.15. *It takes $O(km + 9.5^k\, k^2 n)$ time to solve* Maximum $(m - k)$-Edge Subgraph.

3.4. FPT by random separation

In this section, we use the newly developed random separation method of Cai *et al.* [5] to obtain an FPT algorithm for Minimum $(m - k)$-Edge Domination.

The main idea of the random separation method is to use a random partition of V to separate a solution from the rest of the graph into connected components and then select appropriate components to form a solution. We can use universal sets to derandomize algorithms obtained from random separation. Recall that a list of binary vectors of length n is (n, t)-*universal* if for every subset of size t of the indices, all 2^t configurations appear in the list.

We note that Cai et al. [5] have also used random separation to obtain FPT algorithms for the following four problems in Table 1: MINIMUM $(m - k)$-EDGE COVER, MINIMUM $(m - k)$-EDGE SUBGRAPH, MINIMUM $(m - k)$-EDGE LINK and MAXIMUM $(m - k)$-EDGE LINK.

3.4.1. *Minimum $(m - k)$-edge domination*

We now use the random separation method to solve MINIMUM $(m - k)$-EDGE DOMINATION. In order for $m - k$ edges E' to be adjacent to the minimum number of edges in $E - E'$, we need only maximize the number of edges in $E - E'$ that are not adjacent to E'. Recall that for a set V' of vertices, $e(V')$ denotes the number of edges covered by vertices in V'. Without loss of generality, we may assume that the input graph G has no isolated vertices.

LEMMA 3.16. *Let V^* be a set of vertices that maximizes the number of edges in $G[V^*]$ subject to $e(V^*) \leq k$. Then every subset of $m - k$ edges in $G - V^*$ is an optimal solution of G for* MinED$(m - k)$.

Proof. Let m^* be the number of edges in $G[V^*]$, E' an arbitrary subset of $m - k$ edges in G. Since $e(V - V(E')) \leq k$, $G - V(E')$ has at most m^* edges and thus $|N(E')| \geq k - m^*$.

On the other hand, $|N[E(G - V^*)]| \leq m - m^*$. Therefore, for an arbitrary subset $\tilde{E}$ of $m - k$ edges from $G - V^*$, we have $|N[\tilde{E}]| \leq m - m^*$ as $\tilde{E} \subseteq E(G - V^*)$. Since $|N(\tilde{E})| = |N[\tilde{E}]| - (m - k)$, we have $|N(\tilde{E})| \leq k - m^*$, and therefore $\tilde{E}$ is an optimal solution. □

Therefore, to solve MinED$(m - k)$, we need only find a V^* in Lemma 3.16, and then arbitrarily return $m - k$ edges in $G - V^*$ as an optimal solution. Since V^* covers at most k edges, $V^* \cup N_G(V^*)$ contains at most $2k$ vertices. This property of V^* enables us to find it and thus solve MinED $(m - k)$ in FPT time using the random separation method.

First, we colour each vertex of G randomly and independently by either green or red (each with probability 1/2) to form a random partition of V into green vertices V_g and red vertices V_r, which forms the *green subgraph* $G_g = G[V_g]$, whose components are called *green components*.

A partition of V is a 'good partition' if there is a V^* in Lemma 3.16 such that all vertices in V^* are green and all vertices in the open neighbourhood $N_G(V^*)$ are red. Since $V^* \cup N_G(V^*)$ contains at most $2k$ vertices, the probability that a random partition is a good partition is at least $2^{-2k} = 4^{-k}$, and thus with probability at least 4^{-k}, $G[V^*]$ consists of a collection green components of G_g.

To find an optimal solution from a good partition, we first compute, for each green component H_i, the number m_i of edges in H_i and the number e_i of edges covered by vertices in H_i. This can be easily done in $O(m + n)$ time. Note that for any two green components H_i and H_j, the number of edges in $H_i \cup H_j$ equals $m_i + m_j$ and the number of edges covered by vertices in $H_i \cup H_j$ equals $e_i + e_j$. Therefore, V^* is a collection $\mathcal{H}$ of green components that maximizes

$$\sum_{H_i \in \mathcal{H}} m_i$$

subject to $\sum_{H_i \in \mathcal{H}} e_i \leq k$. This is a $0-1$ knapsack problem and can be solved in $O(kn)$ time by dynamic programming [15]. Therefore, with probability at least 4^{-k} we can find a V^* in $O(m + kn)$ time.

To derandomize the algorithm, we can use a family of partitions such that for every partition P of any $2k$ vertices, there is a partition in the family that is consistent with P. Clearly, such a family is just a family of $(n, 2k)$-universal sets. Naor *et al.* [16] have a deterministic construction for (n, t)-universal sets of size $2^t t^{O(\log t)} \log n$ which can be listed in linear time. Therefore, we obtain a deterministic algorithm that runs in $4^k k^{O(\log k)}(m + kn)\log n$ time.

THEOREM 3.17. *It takes $4^k k^{O(\log k)}(m + kn)\log n$ time to find a set V^* of vertices in a graph G that maximizes the number of edges in $G[V^*]$ subject to the constraint that V^* covers at most k edges.*

COROLLARY 3.18. *It takes $4^k k^{O(\log k)}(m + kn)\log n$ time to solve* MINIMUM $(m - k)$-EDGE DOMINATION.

4. EXACT ALGORITHMS

In this section, we present new exact algorithms for $W[1]$-hard fixed-cardinality optimization problems in connection with the number of edges incident with k vertices, i.e. k-VERTEX COVER, k-VERTEX SUBGRAPH and $(k, n - k)$-CUT problems. Our algorithms use matrix multiplication and run in about $O(n^{0.8k})$ time, improving $\Omega(n^k)$-time exhaustive search algorithms. Note that it takes $O(n^{\omega})$ time, where $\omega < 2.376$, to compute the product of two n by n matrices [17].

The main idea of our algorithms is a generalization of the following $O(n^{\omega t})$ algorithm of Nešetřil and Poljak [18] for finding a $3t$-clique in a graph: construct an auxiliary graph $G' = (V', E')$ such that each vertex in V' is a t-clique in G and two vertices $X, Y \in V'$ are adjacent in G' iff $X \cup Y$ is a $2t$-clique in G, and then use matrix multiplication to find a triangle in G', which corresponds to a $3t$-clique in G. To solve our problems, we transform them into problems of finding appropriate triangles in weighted graphs.

We will start with a discussion on algorithms to find a triangle of maximum (minimum) weight or exact weight in Section 4.1. Then we give algorithms for k-Vertex Cover problems in Section 4.2, and present similar algorithms for k-Vertex Subgraph and $(k,\ n-k)$-Cut problems in Section 4.3. We note that our method in this section can also be used to solve Weight-k 2-Satisfiability problems [19] and we expect it to be useful for other problems as well.

4.1. Maximum and minimum triangles

Let $G=(V, E; w)$ be a weighted graph with $w : V \cup E \to Z$. The weight of a triangle equals the sum of weights of all vertices and edges in the triangle, and a *maximum triangle* (respectively, *minimum triangle*) in G is a triangle of maximum weight (minimum weight). Recall that $\tilde{O}(f(n))$ is a shorthand of $O(f(n)\log^c n)$, where c is a constant.

Theorem 4.1. *A maximum (minimum) triangle in a weighted graph $G=(V, E; w)$ with w: $V \cup E \to \{-W, \ldots, 0, \ldots, W\}$ can be found in $\tilde{O}(Wn^{\omega})$ time.*

Proof. Since T is a maximum triangle in G iff T is a minimum triangle when the weight of each edge and vertex is changed to its negation, we need only consider the problem of finding a minimum triangle.

First, we shift all vertex weights to edges. Define a new weighted graph $G'=(V, E; w')$ with w': $E \to Z$ by setting $w'(uv)=2w(uv)+w(u)+w(v)$ for every edge uv. The weight of a triangle in G' is the sum of weights of all edges in the triangle. Then for any triangle T we have $w'(T)=2w(T)$, and therefore T is a minimum triangle in G iff it is a minimum triangle in G'.

To find a minimum triangle in G', we relate our problem to the distance product of matrices. Let A and B be two $n \times n$ matrices. The *distance product* (a.k.a. *min/plus product*) of A and B, denoted $A \bigstar B$, is the $n \times n$ matrix $C=[c_{ij}]$ with $c_{i,j}=\min_{s=1}^{n}\{a_{is}+b_{sj}\}$. Let $V=\{1, \ldots, n\}$ and define an $n \times n$ matrix D as follows:

$$D[i,j]=\begin{cases} w'(i,j) & \text{if}(i,j)\in E \\ +\infty & \text{otherwise.}\end{cases}$$

Note that $D[i, i]=+\infty$ for each i.

To find the weight of a minimum triangle in G, we compute $T=D \bigstar D+D$. It is clear that entry t_{ij} of T equals the minimum weight of triangles containing edge (i, j), and a minimum entry $t_{i'j'}$ of T equals the weight of a minimum triangle of G. By a result of Alon *et al.* [20], we can compute $D \bigstar D$ and hence T in $\tilde{O}(Wn^{\omega})$ time (note that edge weights of G' are in $\{-4W, \ldots, 0, \ldots, 4W\}$). To find a minimum triangle, we compute in $O(n)$ time a minimum weight 2-path between vertices i' and j', which forms a minimum triangle with edge $i'j'$. Therefore, the overall time is $\tilde{O}(Wn^{\omega})$. □

We also have the following result for finding a triangle of exact weight.

Theorem 4.2. *A triangle of weight exactly l in a weighted graph $G=(V, E; w)$ with w: $V \cup E \to \{-W, \ldots, 0, \ldots, W\}$ can be found in $O(W^2 n^{\omega})$ time.*

Proof. By the argument in the proof of Theorem 4.1, we need only consider edge-weighted graphs with edge weights in $\{-4W, \ldots, 0, \ldots, 4W\}$. Let $V=\{1, \ldots, n\}$. A triple (w_1, w_2, w_3), where each $w_i \in \{-4W, \ldots, 0, \ldots, 4W\}$, is an *$l$-triple* if

$$w_1+w_2+w_3=l.$$

Note that integers in the triple need not be distinct.

Clearly, G has a triangle of weight l iff there is an l-triple (w_1, w_2, w_3) such that G has a triangle whose three edges have weights w_1, w_2 and w_3, respectively. To find a triangle T of weight l in G for a given l-triple (w_1, w_2, w_3), we construct three unweighted graphs G_i, $1 \le i \le 3$, from G as follows: every vertex of G is a vertex of each G_i, an edge uv of G is an edge of graph G_i iff $w(uv)=w_i$. Let A_i be the adjacency matrix of G_i, and let $B=A_1A_2$. Then G contains a triangle of weight exactly l iff there is an edge uv of G_3 for which $B[u, v]=1$.

Since $w_3=l-w_1-w_2$ for any l-triple (w_1, w_2, w_3), there are at most $(8W+1)^2$ l-triples, which can be listed in $O(W^2)$ time. We note that each adjacency matrix A_i can be obtained in $O(n^2)$ time, and matrix B can be computed in $O(n^{\omega})$ time. Furthermore, once we know that $B[u, v]=1$ for some edge uv of G_3, we can easily find a triangle of weight exactly l in $O(n)$ time by locating a vertex w that is adjacent to u in G_1 and v in G_2. Therefore, it takes $O(W^2n^{\omega})$ time to find a triangle of weight exactly l. □

4.2. Optimal k-vertex covers

We now use the results in the previous section to solve the $W[1]$-hard problem MaxVC(k) of finding a maximum k-vertex cover in $\tilde{O}(kn^{\omega\ \lfloor k/3\rfloor+1+k \bmod 3})$ time. The idea is to transform MaxVC(k) into the problem of finding a maximum triangle in an auxiliary graph. We will also use this idea to solve other k-vertex cover problems in a similar amount of time.

Recall that for a set X of vertices, $e(X)$ denotes the number of edges covered by vertices in X; and for two disjoint sets X, Y of vertices, $e[X, Y]$ denotes the number of edges with one end in X and the other end in Y.

First, let us assume that $k=3t$ for some integer t. To find a maximum k-vertex cover in G, we transform the problem to the problem of finding a maximum triangle in the following *t-subset auxiliary graph* $G'=(V', E'; w)$: every subset X of t vertices in G is a vertex in G', and two vertices $X, Y \in V'$ are adjacent in G' iff X and Y are disjoint. The weight function w: $V' \cup E' \to Z$ is defined by $w(X)=e(X)$ for each vertex $X \in V'$ and $w(XY)=-e[X, Y]$ for each edge $XY \in E'$.

LEMMA 4.3. *Graph G has $3t$ vertices covering l edges iff its t-subset auxiliary graph G' has a triangle with weight l.*

Proof. Let X, Y, Z be three mutually disjoint t-subsets of vertices in G, and let $S = X \cup Y \cup Z$. Then $X, Y, Z \in V'$ and they form a triangle T in G'. In graph G, the number of edges covered by S equals

$$e(X) + e(Y) + e(Z) - e[X, Y] - e[Y, Z] - e[Z, X],$$

which exactly equals $w(T)$, the sum of weights of vertices and edges in T. □

Now for a general k, let $t = \lfloor k/3 \rfloor$ and $r = k \bmod 3$. To find a maximum k-vertex cover in G, we find, for every r-subset V' of vertices, a maximum $3t$-vertex cover in $G - V'$. Combining with Lemma 4.3, we obtain the following algorithm for finding a maximum k-vertex cover in a graph G.

```
Algorithm MaxVC(k)
Input: Graph G with at least k vertices.
$t := \lfloor k/3 \rfloor$ and $r := k \bmod 3$;
$S := \varnothing$ and $l := 0$;
for every r-subset V' of vertices in G do
  {Find a maximum 3t-vertex cover S' in $G - V'$.}
  Construct the t-subset auxiliary graph G' of $G - V'$;
  Find a maximum triangle T in G';
  Let S' be vertices in $G - V'$ that correspond to vertices in T;
(*) if $l < w(T) + e(V')$ then $l := w(T) + e(V')$ and
                              $S := V' \cup S'$;
end for;
output S.
```

THEOREM 4.4. *Algorithm* **MaxVC**(k) solves MaxVC(k) in $\tilde{O}(kn^{\omega \lfloor k/3 \rfloor + 1 + k \bmod 3})$ *time.*

Proof. Clearly, every maximum k-vertex cover of G consists of an r-subset V' of vertices and a maximum $3t$-vertex cover of $G - V'$. By Lemma 4.3, the algorithm correctly finds a maximum $3t$-vertex cover S' in $G - V'$. Since the algorithm considers all possible r-subsets V' and returns a set $S = V' \cup S'$ that maximizes the number of edges covered by the set, S is indeed a maximum k-vertex cover of G.

For the complexity of the algorithm, we first determine the running time for each iteration of the 'for' loop. It is easy to see that the construction of G' takes $O(t^2 n^{2t})$ time, as G' has $O(n^t)$ vertices and the weight of each vertex or edge in G' can be computed in $O(t^2)$ time. Since the maximum absolute value of weights of vertices or edges in G' is at most tn, it follows from Theorem 4.1 that a maximum triangle in G' can be found in $\tilde{O}(tnn^{\omega t})$ time. Therefore, each iteration of the 'for' loop takes $\tilde{O}(tn^{\omega t+1})$ time. Since the 'for' loop is executed $O(n^r)$ times and $t = \lfloor k/3 \rfloor$, we have the claimed complexity for the algorithm. □

COROLLARY 4.5. *It takes* $\tilde{O}(kn^{\omega \lfloor k/3 \rfloor + 1 + k \bmod 3})$ *time to solve* MinVC(k), MinVS($n - k$) *and* MaxVS($n - k$).

Proof. MinVC(k) for G is equivalent to MaxVC(k) for the complement of G, MinVS($n - k$) is equivalent to MaxVC(k), and MaxVS($n - k$) is equivalent to MinVC(k). □

Similarly, we can use Lemma 4.3 to obtain the following algorithm that finds k vertices in G, if G has such k vertices, that cover exactly a given number l of edges.

```
Algorithm XctVC (k)
Input: Graph G with at least k vertices, and positive integer l.
$t := \lfloor k/3 \rfloor$ and $r := k \bmod 3$;
$S := \varnothing$;
for every r-subset V' of vertices in G do
  {Find 3t vertices S' in $G - V'$ that cover exactly $l - e(V')$
   edges.}
  Construct the t-subset auxiliary graph G' of $G - V'$;
  if G' contains a triangle T of weight $l - e(V')$
    then let S' be vertices in $G - V'$ that correspond to
           vertices in T;
      output $V' \cup S'$ and stop
  end if
end for;
output 'No such k-vertex cover'.
```

THEOREM 4.6. *Algorithm* **XctVC**(k) solves EXACT k-VERTEX COVER in $O(k^2 n^{\omega \lfloor k/3 \rfloor + 2 + k \bmod 3})$ time.

Proof. The correctness of the algorithm follows from Lemma 4.3. The complexity analysis is the same as Theorem 4.4 except that, by Theorem 4.2, it takes $O(k^2 n^{\omega \lfloor k/3 \rfloor + 2})$ time to find a triangle of weight exactly $l - e(V')$. □

Since a graph has k vertices V' that cover exactly l edges iff $G[V - V']$ contains exactly $m - l$ edges, the following result follows directly from the above theorem.

COROLLARY 4.7. *It takes* $O(k^2 n^{\omega \lfloor k/3 \rfloor + 2 + k \bmod 3})$ *time to solve* EXACT ($n - k$)-VERTEX SUBGRAPH.

4.3. Optimal k-vertex subgraphs and cuts

We now use the ideas in the previous section to solve k-VERTEX SUBGRAPH and (k, $n - k$)-CUT problems. To solve MaxVS(k), we use algorithm **MaxVC**(k) with the following two changes:

(i) For the t-subset auxiliary graph G' of graph $G - V'$, set vertex weight

$$w(X) = e[X, V'] + |E(G[X])|$$

and edge weight $w(XY) = e[X, Y]$.

(ii) In line (*), replace $w(T) + e(V')$ by $w(T) + |E(G[V'])|$.

Theorem 4.8. *It takes* $\tilde{O}(k^2 n^{\omega\lfloor k/3\rfloor + k \bmod 3})$ *time to solve* MaxVS(k).

Proof. Let S be a set of k vertices in G. Then S can be written as $X \cup Y \cup Z \cup V'$, where X, Y and Z are mutually disjoint t-subsets of vertices and $V' = S - (X \cup Y \cup Z)$ which has r vertices. Therefore, X, Y, Z form a triangle T in G' and the number of edges in $G[S]$ equals

$$\begin{aligned}
&(e[X, V'] + |E(G[X])|) + (e[Y, V'] + |E(G[Y])|) + (e[Z, V'] \\
&\quad + |E(G[Z])|) + e[X, Y] + e[Y, Z] + e[Z, X] + |E(G[V'])| \\
&= w(X) + w(Y) + w(Z) \\
&\quad + w(XY) + w(YZ) + w(ZX) + |E(G[V'])| \\
&= w(T) + |E(G[V'])|.
\end{aligned}$$

This proves the correctness of the algorithm.

For the running time of the algorithm, we note that an induced subgraph on k vertices contains at most $\binom{k}{2}$ edges and thus the maximum absolute value W of vertex or edge weights in G' is $O(k^2)$. By an argument similar to that for Theorem 4.4, we have the claimed time bound. □

Corollary 4.9. *It takes* $\tilde{O}(k^2 n^{\omega\lfloor k/3\rfloor + k \bmod 3})$ *time to solve* MinVS(k), MinVC($n - k$) *and* MaxVC($n - k$).

Proof. MinVS(k) for G is equivalent to MaxVS(k) for the complement of G, MinVC($n - k$) is equivalent to MaxVS(k), and MaxVC($n - k$) is equivalent to MinVS(k). □

We can make similar modifications to Algorithm **XctVC**(k) to find k vertices to induce a subgraph with exactly a given number of edges (equivalently, $n - k$ vertices to cover exactly a given number of edges).

Theorem 4.10. *It takes* $O(k^4 n^{\omega\lfloor k/3\rfloor + k \bmod 3})$ *time to solve* Exact k-Vertex Subgraph *and* Exact ($n - k$)-Vertex Cover.

To solve MaxCut(k), we use algorithm **MaxVC**(k) with the following two changes:

(i) For the t-subset auxiliary graph G' of $G - V'$, set vertex weight $w(X) = e[X, V - X] - 2e[X, V']$ and edge weight $w(XY) = -2e[X, Y]$.
(ii) In line (*), replace $w(T) + e(V')$ by $w(T) + e[V', V - V']$.

Theorem 4.11. *It takes* $\tilde{O}(kn^{\omega\lfloor k/3\rfloor + 1 + k \bmod 3})$ time to solve MaxCut(k).

Proof. A set S of k vertices in G can be written as $X \cup Y \cup Z \cup V'$, where X, Y and Z are mutually disjoint t-subsets of vertices and $V' = S - (X \cup Y \cup Z)$, which has r vertices. Therefore, X, Y, Z form a triangle T in G' and the number of edges in the cut $[S, V - S]$ can be expressed as

$$\begin{aligned}
&(e[X,V-X]-e[X,V']-e[X,Y]-e[X,Z]) \\
&\quad +(e[Y,V-Y]-e[Y,V']-e[Y,Z]-e[Y,X]) \\
&\quad +(e[Z,V-Z]-e[Z,V']-e[Z,X]-e[Z,Y]) \\
&\quad +(e[V',V-V']-e[X,V']-e[Y,V']-e[Z,V']) \\
&\quad =w(X)+w(Y)+w(Z)+w(XY)+w(YZ)+w(ZX)+e[V',V-V'] \\
&\quad =w(T)+e[V',V-V'].
\end{aligned}$$

This proves the correctness of the algorithm.

For the running time of the algorithm, we note that there are at most kn edges incident with k vertices and thus the maximum absolute value W of vertex or edge weights in G' is $O(kn)$. By an argument similar to that for Theorem 4.4, we obtain the claimed complexity. □

Since MinCut(k) for G is equivalent to MaxCut(k) for the complement of G, the above theorem implies the following result.

Corollary 4.12. *It takes* $\tilde{O}(kn^{\omega\lfloor k/3\rfloor + 1 + k \bmod 3})$ *time to solve* MinCut(k).

Again we can make similar modifications to algorithm **XctVC**(k) to find a (k, $n - k$)-cut with exactly a given number of edges.

Theorem 4.13. *It takes* $\tilde{O}(k^2 n^{\omega\lfloor k/3\rfloor + 2 + k \bmod 3})$ *time to solve* Exact (k, $n - k$)-Cut.

5. CONCLUDING REMARKS

We have demonstrated through a large number of graph problems that parameterized complexity analysis is a useful, fruitful and also natural framework for studying cardinality constrained optimization problems. This framework not only complements the classical complexity theory, but may also provide us with new insight into such problems.

As expected, we have seen that many fixed-cardinality optimization problems remain hard under the framework of parameterized complexity. Nevertheless, it is rather surprising that many different graph problems are closely related through their fixed-cardinality optimization problems, and such connections are revealed only through parametric reductions among these problems. Close connections are also evidenced by the central role of k-Clique on regular graphs in our $W[1]$-hardness proofs, and the simplicity of most of our parametric reductions.

In terms of fixed-parameter tractability, we have used various techniques to obtain FPT algorithms. This is certainly encouraging as fixed-cardinality optimization problems are usually very difficult to solve. On the other hand, we need new tools to deal with fixed-cardinality optimization problems more effectively. In this direction, we note that the newly developed random separation method of Cai *et al.* [5] is quite useful and its power seems yet to be fully explored. Furthermore, Cai and Leung [21] have recently developed an *extension method* to solve MAXIMUM k-VERTEX COVER on planar graphs. Their basis idea is to extend iteratively a maximum i-vertex cover to a maximum $(i+1)$-vertex cover, but bound the search space to a small graph. It will be also quite interesting to see how the colour coding method of Alon *et al.* [22], and the iterative compression method of Reed *et al.* [23] can be applied to solve the fixed-cardinality optimization problems.

We have illustrated through several $W[1]$-hard fixed-cardinality graph problems that we can reduce the main cost of their exact algorithms from $O(n^k)$ to around $O(n^{0.8k})$ by transforming them into problems of finding appropriate triangles in auxiliary graphs. It will be interesting to come up with some sufficient conditions or necessary conditions for problems that can be solved by this approach. Of course, it is also interesting to determine whether it is possible to find a maximum weight triangle in $O(n^{\omega})$ time.

For the future directions, we can certainly investigate the parameterized complexity of other cardinality constrained optimization problems. In particular, problems for digraphs seem to be very good candidates. Another important and interesting direction is to study the connections between fixed-parameter tractability and various approximation schemes for cardinality constrained optimization problems. My recent work [24] has indeed revealed some close connections, and I believe that there are deep connections yet to be discovered.

As for the two open problems MINIMUM k-EDGE SUBGRAPH and MAXIMUM k-EDGE MULTICOMPONENT CUT in Table 1, we note that MinES(k) is closely related to the problem of finding a subset of vertices that induces a subgraph with exactly k edges, and remind the reader that MaxEMC(k) is connected to the well-studied multiway cut problem. I conjecture that MinES(k) is FPT, but I am uncertain about MaxEMC(k).

It seems that if we confine ourselves to the P versus NPc question, we actually conceal, rather than reveal, the intricate complexity of cardinality constrained optimization problems. Through the lens of parameterized complexity, we see a colourful world of the complexity of cardinality constrained optimization problems. In my view, parameterized complexity analysis is a natural and integrated part in the study of cardinality constrained optimization problems, and I am looking forward to seeing more exciting results.

ACKNOWLEDGEMENTS

The author is grateful to Michael Fellows, Jianer Chan, Venkatesh Raman, Marcos Cesati, and many other participants in the 1st Workshop on Parameterized Complexity held in Chennai, India, 2000, for fruitful discussions on the parameterized complexity of cardinality constrained optimization problems, and also to a referee for constructive suggestions.

FUNDING

Partially supported by grant CUHK4410/99E from the Reseach Grants Council of Hong Kong SAR.

REFERENCES

[1] Bruglieri, M., Ehrgott, M., Hamacher, H.W. and Maffioli, F. (2006) An annotated bibliography of combinatorial optimization problems with fixed cardinality constraints. *Discrete Appl. Math.*, **154**, 1344–1357.

[2] Downey, R.G. and Fellows, M.R. (1999) Parameterized Complexity. *Springer.*

[3] Downey, R.G., Estivill-Castro, V., Fellows, M.R., Prieto, E. and Rosamund, F.A. (2003) Cutting up is hard to do. Electronic Notes in Theoretical Computer Science, **78**.

[4] Marx, D. (2004) Parameterized graph separation problems. *IWPEC 2004*. Lecture Notes in Computer Science, **3162**, pp. 71–82.

[5] Cai, L., Chan, S.M. and Chan, S.O. (2007) Random separation. Manuscript. (Extended abstract in IWPEC 2006, LNCS 4169 (2006) pp. 239–250 with title 'Random separation: a new method for solving fixed-cardinality optimization problems').

[6] Flum, J. and Grohe, M. (2006) *Parameterized Complexity Theory*. Springer.

[7] West, D.B. (2001) *Introduction to Graph Theory*. Prentice Hall.

[8] Downey, R.G. and Fellows, M.R. (1995) Fixed-parameter tractability and completeness II: On completeness for $W[1]$. *Theoret. Comput. Sci.*, **141**, 109–131.

[9] Chen, J., Kanj, I. and Xia, G. (2006) Improved parameterized upper bounds for vertex cover. Lecture Notes in Computer Science, **4162**, pp. 238–249.

[10] Asahiro, Y., Hassin, R. and Iwama, K. (2002) Complexity of finding dense subgraphs. *Discrete Appl. Math.*, **121**, 15–26.

[11] Garey, M.R. and Johnson, D.S. (1979) *Computers and Intractability: A Guide to the Theory of NP-completeness*. W.H. Freeman, San Francisco.

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

[13] Micali, S. and Vazirani, V.V. (1980) An $O(\sqrt{V \cdot E})$ algorithm for finding maximum matching in general graphs. *Proc. 21st Ann. IEEE Symp. Foundations of Computer Science*, Syracuse, pp. 17–27.

[14] Ore, O. (1962) *Theory of Graphs*. Amer. Math. Soc. Colloq. Publ., 38. Amer. Math. Soc., Providence, RI.

[15] Kleinberg, J. and Tardos, E. (2005) *Algorithm Design*. Pearson.

[16] Naor, M., Schulman, L.J. and Srinivasan, A. (1995) Splitters and near-optimal derandomization. *Proc. 36th Ann. Symp. Foundations of Computer Science*, Los Alamitos, California, USA, pp. 182–191.

[17] Coppersmith, D. and Winograd, S. (1990) Matrix multiplication via arithmetic progressions. *J. Symb. Comput.* **9**, 251–280.

[18] Nešetřil, J. and Poljak, S. (1985) On the complexity of the subgraph problem. *Comment. Math. Univ. Carolinae*, **14**, 415–419.

[19] Cai, L. (2007) Exact and FPT algorithms for weight-k 2-satisfiability problems. Manuscript.

[20] Alon, N., Galil, Z. and Margalit, O. (1997) On the exponent of the all pairs shortest path problem. *J. Comput. Syst. Sci.*, **54**, 255–262.

[21] Cai, L. and Leung, C.W. (2007) The maximum k-vertex cover problem for planar graphs. Manuscript.

[22] Alon, N., Yuster, R. and Zwick, U. (1995) Color-coding. *J. ACM*, **42**, 844–856.

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

[24] Cai, L. (2007) Fixed-parameter tractability and approximation schemes for cardinality constrained optimization problems. Manuscript.