# Parameterized Counting Algorithms for General Graph Covering Problems*

Naomi Nishimura[1], Prabhakar Ragde[1], and Dimitrios M. Thilikos[2]

[1] School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, N2L 3G1

[2] Departament de Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya, Campus Nord, Desp. $\Omega$-228, c/Jordi Girona Salgado, 1-3. E-08034, Barcelona, Spain

**Abstract.** We examine the general problem of covering graphs by graphs: given a graph $G$, a collection $\mathcal{P}$ of graphs each on at most $p$ vertices, and an integer $r$, is there a collection $\mathcal{C}$ of subgraphs of $G$, each belonging to $\mathcal{P}$, such that the removal of the graphs in $\mathcal{C}$ from $G$ creates a graph none of whose components have more than $r$ vertices? We can also require that the graphs in $\mathcal{C}$ be disjoint (forming a "matching"). This framework generalizes vertex cover, edge dominating set, and minimal maximum matching. In this paper, we examine the parameterized complexity of the counting version of the above general problem. In particular, we show how to count the solutions of size at most $k$ of the covering and matching problems in time $O(n \cdot r(pk+r)+2^{f(k,p,r)})$, where $n$ is the number of vertices in $G$ and $f$ is a simple polynomial. In order to achieve the additive relation between the polynomial and the non-polynomial parts of the time complexity of our algorithms, we use the compactor technique, the counting analogue of kernelization for parameterized decision problems.

## 1 Introduction

Parameterized algorithms offer an approach to solving NP-hard problems through the observation that many such problems come with one or more natural parameters which may be small in practice, and so algorithms that are polynomial in the input size but exponential in the parameters may be of practical use. The considerable literature on parameterized complexity provides both algorithms for certain problems (e.g. vertex cover, where the parameter $k$ is the size of the cover) and evidence (in the form of completeness results) that other problems (e.g. clique) do not have efficient parameterized algorithms.

One common technique in designing parameterized algorithms is to find a problem kernel. This consists of reducing an instance of a problem to a smaller

* The two first authors were supported by the Natural Sciences and Engineering Research Council of Canada (NSERC). The third author was supported by the Spanish CICYT project TIN-2004-07925 (GRAMMARS).

instance of size dependent only on the parameters, such that the smaller instance has a solution if and only if the original instance does. Inefficient algorithms (e.g. brute force search) can then be used on the kernel. While this approach is appealing, it may be difficult to find a kernel for a given problem.

Our focus in this paper is on counting the number of solutions constrained by the parameters (e.g. the number of vertex covers of size at most $k$), as first considered by Flum and Grohe [FG04]. Fernau [Fer02] defines fixed-parameter enumerability, and considers the two problems of enumerating all solutions (producing each one, as opposed to counting the total), and of enumerating all optimal solutions. But for many problems (e.g. vertex cover) enumerating all solutions is not fixed-parameter enumerable, as there are too many solutions. This naturally suggests our approach of counting the solutions without enumerating them.

We consider a different sort of kernel-like structure from that used in traditional parameterized algorithms, one specialized for counting. Such a kernel comes with a function mapping a solution in the original instance to one in the kernel, in a fashion that allows us to compute the size of each preimage. That way, we reduce the problem of counting the solutions of the original problem to the problem of enumerating the solutions in the kernel. We count solutions in the original instance by using a (possibly inefficient) algorithm to enumerate solutions in the kernel and summing the sizes of preimages. This method was used by Díaz, Serna, and Thilikos [DST04a, DST04b] in the context of colouring problems; here we apply it to covering and matching problems. This method is a departure from previous work on parameterized counting [Fer02, Dam04, AR02, DST04b]. Our goal is to obtain running times with an additive relation between the part that is polynomial in the input size and the part that is possibly exponential in the parameters.

Our problems are defined by a graph $G$, a collection $\mathcal{P}$ of graphs each on at most $p$ vertices, an integer $r$, and a parameter $k$. We wish to "cover" $G$ by $k$ graphs chosen from $\mathcal{P}$, such that the connected subgraphs left uncovered have no more than $r$ vertices each. That is, we ask whether or not there is a collection $\mathcal{C}$ of subgraphs of $G$ (with $|\mathcal{C}| \leq k$), each belonging to $\mathcal{P}$, such that the removal of the graphs in $\mathcal{C}$ from $G$ creates a graph none of whose components have more than $r$ vertices. In this formulation, we allow the graphs in $\mathcal{C}$ to overlap, forming a "covering". Another variation of the problem requires that the graphs in $\mathcal{C}$ be disjoint, forming a "matching".

This framework generalizes vertex cover, edge dominating set, and minimal maximum matching. For vertex cover, $\mathcal{P}$ contains only the graph with one vertex, and $r = 1$; for edge dominating set, $\mathcal{P}$ contains only the graph with two connected vertices, and $r = 1$; for minimal maximum matching, we add the constraint that the graphs in $\mathcal{C}$ must be disjoint. Interestingly, minimum maximal maximal matching and edge dominating set are polynomially equivalent as decision problems, but not as counting problems. In this paper, we show how to count the number of solutions of the general covering and matching problems problems in time $O(n \cdot r(pk + r) + 2^{O(pkr(pk+r))})$ where $n$ is the number of vertices in $G$.

## 2 Basic Definitions

All graphs in this paper are undirected, loopless and without multiple edges. For each graph $G$ considered, we will denote as $V(G)$ and $E(G)$ its vertex and edge set, respectively. Given a set $S \subseteq V(G)$ we define $N_G(S)$ as the set of all vertices $v \in V(G) - S$ such that $v$ has a neighbour in $S$. For a set of graphs $\mathcal{C}$, we denote as $\mathbf{V}(\mathcal{C})$ the set of all vertices of the graphs in $\mathcal{C}$, i.e. $\mathbf{V}(\mathcal{C}) = \bigcup_{G \in \mathcal{C}} V(G)$.

For $p$ a fixed constant and $\mathcal{P}$ a fixed set of graphs of no more than $p$ vertices, we define the following parameterized problems:

$(k,r)$-Minimum Covering by Graphs in $\mathcal{P}$ ($(k,r)$-$\mathsf{MCG}$-$(\mathcal{P})$)
*Input:* A graph $G$, a collection of graphs $\mathcal{P}$, and an integer $r$.
*Parameter:* A non-negative integer $k$.
*Question:* Does $G$ contain a collection $\mathcal{C}$ of $k$ subgraphs each isomorphic to some graph in $\mathcal{P}$ and such that $G[V(G) - \mathbf{V}(\mathcal{C}))]$ has no component of size more than $r$?

If in the above problem we demand that the graphs in $\mathcal{C}$ be pairwise disjoint (i.e. no vertices in common) then we define the $(k,r)$-Minimum Maximal Matching by Graphs in $\mathcal{P}$ ($(k,r)$-$\mathsf{MMM}$-$(\mathcal{P})$).

We denote by $\mathbf{MCG}_k(G)$ the set of solutions of the $(k,r)$-$\mathsf{MCG}$-$(\mathcal{P})$ problem when the input is $G$ and the parameter is $k$. Similarly, we define $\mathbf{MMM}_k(G)$ and notice that $\mathbf{MCG}_k(G) \subseteq \mathbf{MMM}_k(G)$. Also we define $\mathsf{mcg}_k(G) = |\mathbf{MCG}_k(G)|$ and $\mathsf{mmm}_k(G) = |\mathbf{MMM}_k(G)|$.

In what follows we will give a parameterized counting algorithm for each of the above problems. In particular, we will give two algorithms that output $\mathsf{mcg}_k(G)$ and $\mathsf{mmm}_k(G)$, respectively, in $O(n \cdot r(pk+r) + 2^{O(pkr(pk+r))})$.

A basic tool for our algorithms is the notion of $(a,b)$-central set. In particular, we say that a subset $P$ of $V(G)$ is an $(a,b)$*-central set* (or *central set* for short) if $|P| \leq a$ and each connected component of $G[V(G) - P]$ has at most $b$ vertices. Notice that a vertex cover of size at most $k$ is a $(k,1)$-central set and vice versa. For convenience, we refer to the connected components of $G[V(G) - P]$ as *satellite graphs*. Of particular interest are those satellite graphs that have neighbours in $P$; we will call these *dependent* satellite graphs and all others *independent* satellite graphs.

To form our counting algorithms, we use the notion of *compactor enumeration* as introduced by Díaz, Serna, and Thilikos [DST04a, DST04b]. The idea is to find a particular kind of kernel for a parameterized problem such that any solution of the problem can be mapped to a solution within the kernel. If we can enumerate all the solutions within the kernel, and for each one, compute (in a reasonable amount of time) the number of preimages of general solutions mapping to it, we can count the number of general solutions of the problem. More formally, a compactor $\mathsf{Cmp}(\Pi, k)$ for a parameterized problem $\Pi$ with parameter $k$ and set of solutions $\mathsf{Sol}(\Pi, k)$ has the following properties: $|\mathsf{Cmp}(\Pi, k)|$ is a function that depends only on $k$; $\mathsf{Cmp}(\Pi, k)$ can be enumerated with an algorithm whose complexity depends only on $k$; there is a surjective function $m : \mathsf{Sol}(\Pi, k) \rightarrow$

$\mathsf{Cmp}(\Pi, k)$; and for any $C \in \mathsf{Cmp}(\Pi, k)$, $|m^{-1}(C)|$ can be computed within time $O(f(k)n^c)$.

## 3 Central Sets

The notion of a central set plays a key role in our algorithms, as a necessary condition for a nonzero number of solutions (Lemma 1) and as an important step towards forming a compactor.

**Lemma 1.** *If for a graph $G$ $\mathsf{mcg}_k(G) > 0$ or $\mathsf{mmm}_k(G) > 0$, then $G$ contains a $(pk, r)$-central set.*

*Proof.* We present the proof for the case $\mathsf{mcg}_k(G) > 0$, as the proof for the case $\mathsf{mmm}_k(G) > 0$ is identical. Since there is at least one solution to the problem, we let $\mathcal{C}$ be one such solution. By definition, the collection $\mathcal{C}$ consists of $k$ subgraphs each isomorphic to a graph in $\mathcal{P}$, and hence the total number of vertices in $\mathbf{V}(\mathcal{C})$ is at most $pk$. Moreover, again by the definition of the problem, $G[V(G) - \mathbf{V}(\mathcal{C}))]$ has no component of size more than $r$. This implies that $\mathcal{C}$ is a $(pk, r)$-central set, as claimed.

As central sets are used in our covering and matching algorithms, the complexity of finding an $(a, b)$-central set for a graph $G$ has an impact on the complexity of our counting algorithms. The problem of determining whether $G$ has an $(a, b)$-central set is NP-hard when $a$ and $b$ are both part of the input, as for $b = 1$ the problem is vertex cover. If $a$ is fixed, the brute-force checking of all $O(n^a)$ candidate solutions constitutes a polynomial-time algorithm. For the case in which $b$ is fixed, the problem can be shown to be NP-hard using a reduction from vertex cover in which $G$ is transformed into a graph $G'$ by attaching an $b$-clique to each vertex $v \in V(G)$: then $G$ has a vertex cover of size $a$ if and only if $G'$ has an $(a, b)$-central set. Our parameterized solution follows.

**Lemma 2.** *An $(a, b)$-central set of a graph $G$ can be found, if it exists, in time $O(n(a + b) + (ab(a + b - 1) + a)(b + 1)^a)$, where $n$ is the number of vertices in $G$; otherwise, in the same time bound it can be determined that $G$ has no $(a, b)$-central set.*

*Proof.* We present an algorithm, **FIND-CENTRAL-SET**$(a, b, G)$, that determines whether $G$ has an $(a, b)$-central set and, if so, returns one $(a, b)$-central set. We first observe that if a vertex $v$ of $G$ has degree greater than $a + b - 1$, it must be in the $(a, b)$-central set $C$, as otherwise the placement of any $a$ of its neighbours in $C$ would leave a graph of size at least $b+1$ in $G[V(G)-C]$ (namely $v$ and its remaining neighbours), violating the definition of an $(a, b)$-central set. Consequently, if there are more than $a$ such high-degree vertices, since the size of the $(a, b)$-central set is at most $a$, we can conclude that we have a NO-instance, as indicated in Step 1 below.

```
FIND-CENTRAL-SET$(a, b, G)$
1. Let $A$ contain all vertices of $G$ that have degree greater than $a+b-1$.
   If $|A| > a$ then return NO.
2. Let $G' = G[V(G) - A]$ and $a' = a - |A|$. Let $G^*$ be the union
   of the connected components of $G'$ that have more than $b$ vertices. If
   $|V(G^*)| > a'b(a+b-1) + a'$ then return NO.
3. If ST-CENTRAL-SET$(a', b, G^*)$ returns NO, return NO and return;
   otherwise, let $C$ be the $(a', b)$-central set of $G^*$ that is returned.
4. return YES and $A \cup C$ as an $(a, b)$-central set of $G$.
```

We can then reduce the problem to that of finding an $(a', b)$-central set in a graph $G^*$ of degree at most $a + b - 1$ in Steps 2 and 3, where $a'$ is $a$ minus the number of high-degree vertices found in the previous paragraph. We observe that if $G' = G[V(G) - A]$ is a YES-instance, there can be at most $a'(a + b - 1)$ dependent satellite graphs, since each of the $a'$ vertices in the $(a', b)$-central set have at most $a + b - 1$ neighbours. As each dependent satellite graph has at most $b$ vertices and the central set has at most $a'$, the size of $G^*$ can be at most $a'b(a + b - 1) + a'$. Having obtained a graph of size dependent only on the parameters $a$ and $b$, it is now possible to obtain a solution using the search-tree based routine **ST-CENTRAL-SET**$(a, b, G)$ below. The routine consists of checking if all connected components are of size at most $b$ for the base case $a = 0$, and otherwise choosing $b + 1$ vertices that share a component and trying each as a possible member of the central set.

We first determine the running time of **ST-CENTRAL-SET**$(a', b, G^*)$, observing that the depth of the recursion will be at most $a'$. We can find connected components in time linear in the size of $G^*$, or in time $O(a'b(a + b - 1) + a')$. In Step 3, the routine is called $b + 1$ times, giving a total running time of $O((a'b(a + b - 1) + a')(b + 1)^{a'}) = O((ab(a + b - 1) + a)(b + 1)^a)$.

```
ST-CENTRAL-SET$(a, b, G)$
1. If $a = 0$, check whether each connected component of $G$ has at most
   $b$ vertices; if so, return YES, and if not, return NO.
2. Let $K$ be any set of $b+1$ vertices inducing a connected subgraph of
   $G$. If no such $K$ can be found, return YES.
3. For each $v \in K$, determine ST-CENTRAL-SET$(a-1, b, G')$ where
   $G' = G[V(G) - \{v\}]$. If any answer is YES, return YES and the set
   of vertices removed in the sequence of calls leading to the YES answer.
   Otherwise, return NO.
```

The running time of **FIND-CENTRAL-SET**$(a, b, G)$ can be determined as follows. Step 1 requires checking at most $a + b$ neighbours of each of the $n$ nodes, in total time $O(n(a + b))$. Determining the connected components of $G'$ and counting the number of vertices in components of size more than $b$ can be completed in linear time, $O(n)$. Thus, using the result above for Step 3, we conclude that the total running time is $O(n(a+b)+(ab(a+b-1)+a)(b+1)^a)$.

## 4 Forming a Compactor

The compactor for $\mathsf{mcg}_k(G)$ is based on the fact that our graph can be viewed as a central set surrounded by satellite graphs. We first group satellite graphs into equivalence classes and then prune the classes to form a reduced graph $G'$. In each counting algorithm, the number of solutions in $G$ is computed by determining the number of solutions in $G$ represented by each solution in $G'$.

To be able to substitute a satellite graph in an equivalence class by another graph in the same class, satellite graphs in the same equivalence class should be isomorphic to each other, have the same neighbourhood in the central set, and have the same attachments to those neighbours. More formally, we first define the graphs formed by the satellites and their neighbours, and then formally define the necessary property. For $H$ a subgraph of $G$, we denote as $\partial_G(H)$ the graph $G[V(H) \cup N_G(V(H))]$. Then, for $G$ a graph and $G_1$ and $G_2$ subgraphs of $G$, we say that $G_1$ and $G_2$ are *friends for* $G$ if the following conditions are satisfied.

1. $N_G(V(G_1)) = N_G(V(G_2))$,
2. There is an isomorphism $\phi$ from $\partial_G(G_1)$ to $\partial_G(G_2)$ where for each $v \in N_G(V(G_1)), \phi(v) = v$.

The counting algorithms proceed by finding a $(pk, r)$-central set, grouping satellites into equivalence classes, pruning the graph $G$ to form a graph $G'$ by reducing the size of each sufficiently large equivalence class, solving the problem on $G'$, and then counting the number of solutions to $G$ represented by the solutions found for $G'$. The pruned graph $G'$ plays the role of the compactor in the formalization in Section 2. Crucial to the algorithm is the formation of $G'$, identifying for each equivalence class $\mathcal{S}$ which graphs are to be retained ($\mathcal{R}_{\mathcal{S}}$) and how many have been omitted ($o_{\mathcal{S}}$).

**Lemma 3.** *Given a graph $G$ and a $(pk, r)$-central set $C$ of $G$, it is possible to determine the following, where $\mathcal{C}$ is the set of connected components of $G[V(G) - C]$, $\mathbf{S}$ is a partition of $\mathcal{C}$ such that any two graphs in the same part are friends, and $\mathbf{S}^*$ is the collection of sets in $\mathbf{S}$ with more than $pk + 1$ graphs:*

$\mathcal{R}_{\mathcal{S}}$ *: a set of $pk + 1$ graphs from $\mathcal{S}$.*
$o_{\mathcal{S}}$ *: the number of graphs that have been omitted from $\mathcal{S}$ to form $\mathcal{R}_{\mathcal{S}}$, namely $|\mathcal{S}| - pk - 1$.*
$G'$ *: the graph formed by removing graphs associated with each $\mathcal{S}$, namely $G' = G[V(G) - \bigcup_{\mathcal{S} \in \mathbf{S}^*} \mathbf{V}(\mathcal{S} - \mathcal{R}_{\mathcal{S}})]$.*

*for each $\mathcal{S} \in \mathbf{S}^*$ in time $O(nr(pk + r) + 2^{r(pk+r)})$, where $|\mathbf{S}^*| \in O(2^{r(pk+r)})$.*

*Proof.* The algorithm **CREATE-KERNEL-SETS**$(p, k, r, C, G)$ partitions satellites into equivalence classes based on an arbitrary ordering on the vertices in the central set $C$ and each satellite in $\mathcal{C}$ (Steps 1 and 2) and a bit vector used to indicate the edges that exist both in the satellite and between the satellite and the central set. In particular, each bit vector has one entry for each potential edge between vertices in the satellite (at most $\binom{r}{2}$ in total) and for each potential edge between a vertex in the satellite and a vertex in the $(pk, r)$-central set (at most $pkr$ in total). Satellites with the same bit vectors will be placed in the same equivalence class.

It is worth noting that the algorithm does not guarantee that friends are in the same part, only that graphs in the same part are friends. This is due to the fact that we are choosing arbitrary orderings of vertices in satellites; if different orderings are chosen, friends will appear in different parts of the partition. We settle for this finer partition in order to realize the goal of having the relation between the polynomial and the exponential parts of the running time be additive.

As we identify the equivalence classes to which satellites belong, we keep track of the number of satellites in each class, retaining the first $pk + 1$ in each class by marking the vertices for inclusion in $G'$ (Step 5). It then follows that $G'$ will consist of all marked vertices and $C$, $\mathcal{R}_{\mathcal{S}}$ will be a set of $pk + 1$ retained satellites, and $o_{\mathcal{S}}$ will be the number of satellites seen minus $pk + 1$ to indicate how many have been omitted.

```
**CREATE-KERNEL-SETS**$(p, k, r, C, G)$
**1**. Arbitrarily label the vertices in $C$ from 1 through $pk$.
**2**. For each component $D \in \mathcal{C}$, arbitrarily label the vertices in $D$ from $pk + 1$ through $pk + |V(D)| \leq pk + r$.
**3**. Create $\sigma$ to map the integers 1 through $d = pkr + \binom{r}{2}$ to the pairs $(i, j)$ where $1 \leq i \leq pk + r$ and $pk + 1 \leq j \leq pk + r$.
**4**. Create an array $R$ of size $2^d$ with each entry storing an integer and a pointer.
**5**. For each component $D$ form a bit vector of size $d$, where entry $\ell$ is set to 1 if and only if $\sigma(\ell) = (i, j)$ such that there is an edge between the vertices with labels $i$ and $j$ in $D$ and (if $i \leq pk$) $C$. Using the value of the bit vector as an index to $R$, increment the entry in $R$; if the entry in $R$ is now at most $pk + 1$, add $D$ to the linked list at $R$ and mark all vertices in $D$.
**6**. Form $o_{\mathcal{S}}$ by subtracting $pk + 1$ from each value in $R$ of size greater than $pk + 1$.
**7**. Create $G'$ by marking all vertices in $C$ and forming the subgraph of $G$ induced on the marked vertices.
**8**. Return as $\mathcal{R}_{\mathcal{S}}$ all linked lists of entries of $R$ with values greater than $pk + 1$, all values $o_{\mathcal{S}}$, and $G'$.
```

To see that the running time is as claimed, we observe that the labels in Steps 1 and 2 can be created in time $O(n)$ for $n$ the number of vertices in $G$, and in Step 3 $\sigma$ can be created in time $O(d) = O(r(pk + r))$. As there are at most $n$ components $D$ and each bit vector is of length $d$, bit vector formation and marking of vertices in Step 5 can be executed in $O(nd)$ time. As there are $2^d$ entries in $R$, Step 6 can be executed in time $O(2^d)$. Finally, since Step 7 will require at most $O(n)$ time, the running time of the algorithm is at most $O(nd + 2^d) = O(nr(pk + r) + 2^{r(pk+r)})$. We observe that $|\mathbf{S}^*| \leq 2^d$ and thus is in $O(2^{r(pk+r)})$.

## 5 Counting Coverings and Matchings

The following theorems present algorithms that make use of the compactor defined in the previous section.

**Theorem 1.** *The value* $\mathsf{mcg}_k(G)$ *can be determined in time*

$$O(n \cdot r(pk+r) + r^{pk}(pk+1)^{pk}2^{pkr(pk+r)}(r(pk+1)2^{r(pk+r)} + 2^{p^2} \cdot (p+3)!)).$$

*Proof.* The algorithm below makes use of the earlier subroutines to find a central set (Step 1) and construct $G'$ by removing all but $pk+1$ remaining satellites in each part of a partition (Step 3); it then finds the solution in $G'$ (Step 4), and counts the number of solutions in $G$ (Step 5).

```
**COMPUTE-mcg-k**$(p, r, G)$
**1**. Use **FIND-CENTRAL-SET**$(pk, r, G)$ to check whether $G$ contains a $(pk, r)$-central set. If the answer is NO then **return** 0.
**2**. Let $C$ be the $(pk, r)$-central set of $G$. Let $\mathcal{C}$ be the set of connected components of $G[V(G) - C]$. Recall that each graph in $\mathcal{C}$ has at most $r$ vertices.
**3**. Use **CREATE-KERNEL-SETS**$(p, k, r, C, G)$ to obtain remaining graphs $\mathcal{R}_{\mathcal{S}}$, numbers of removed graphs $o_{\mathcal{S}}$, and $G'$.
**4**. Compute $\mathbf{MCG}_k(G')$ using brute force.
**5**. Compute and **return** the following number, for $j_{H,\mathcal{S}} = |\{J \in \mathcal{R}_{\mathcal{S}} \mid V(J) \cap V(H) \neq \emptyset\}|$:
```

$$\sum_{\mathcal{G} \in \mathbf{MCG}_k(G')} \prod_{\mathcal{S} \in \mathbf{S}^*} \sum_{\mathcal{G}' \subseteq \mathcal{G}} \prod_{H \in \mathcal{G}'} \binom{j_{H,\mathcal{S}} + o_{\mathcal{S}}}{j_{H,\mathcal{S}}}.$$

The correctness of Step 1 follows from Lemma 1, as any graph $G$ with a non-zero solution will have a $(pk, r)$-central set. In forming $G'$, we need to ensure that in any solution, there is at least one satellite with no vertex in the solution (in essence, representing all the pruned satellites). As any solution will be of at most $k$ graphs of at most $p$ vertices each, the entire solution will consist of at most $pk$ vertices; retaining $pk+1$ satellites will thus satisfy the needed condition.

The correctness of the algorithm depends on the counting in Step 5, summing over each solution $\mathcal{G}$ in the reduced graph the number of solutions in the original graph that are represented by $\mathcal{G}$. In particular, graphs involving remaining satellites (i.e., those in $\mathcal{R}_{\mathcal{S}}$) in a particular equivalence class can be replaced by graphs involving satellites that were omitted from the class to form the reduced graph. To count all such possibilities for a particular solution, we consider one particular pruned equivalence class $\mathcal{S}$, and observe that our total result will be formed by taking the product of all such values; this is because the effects of making such alterations are independent for each equivalence class.

For a fixed solution $\mathcal{G}$ and equivalence class $\mathcal{S}$, we consider all possible ways of exchanging a subset $\mathcal{G}'$ of the collection of graphs forming the solution for satellites that have been omitted. This entails summing over all subsets $\mathcal{G}'$ of graphs in $\mathcal{G}$, and then for each graph $H$ in the subset $\mathcal{G}'$ counting all the ways of making exchanges. The graph $H$ may use more than one satellite in the set $\mathcal{R}_\mathcal{S}$, where a graph $J$ is used by $H$ precisely when it is a remaining graph (and hence in $\mathcal{R}_\mathcal{S}$) and includes at least one vertex of $H$ (and hence $V(J) \cap V(H)$). The number $j_{H,\mathcal{S}}$ of graphs used is thus $j_{H,\mathcal{S}} = |\{J \in \mathcal{R}_\mathcal{S} \mid V(J) \cap V(H) \neq \emptyset\}|$, and we need to count the number of ways to choose $j_{H,\mathcal{S}}$ such graphs out of the ones used plus the set $\mathcal{O}_\mathcal{S}$, in order to count the solutions in the general graph represented by this one.

To see that we are not overcounting or undercounting the number of solutions in the complete graph that are represented by a solution in the reduced graph, we first impose an arbitrary ordering on all graphs in each equivalence class and for each choose an isomorphism as defined in the term *friends*. We now observe that if there is more than one way to map the same part of $H$ to a particular satellite, each of the mappings entails a different solution in the reduced graph, and hence we count these different possibilities by our counting ways of swapping satellites for the other solutions.

To determining the running time of the algorithm, we first observe that due to Lemma 2, the running time of Step 1 is in $O(n(pk + r) + pkr(pk + r - 1) + pk)(r + 1)^{pk})$. Finding connected components in Step 2 will take linear time. The running time of Step 3, $O(nr(pk + r) + 2^{r(pk+r)})$, is a direct consequence of Lemma 3.

A brute-force approach for Step 4 will consist of trying all possible choices of a collection $\mathcal{C}$ of $k$ subgraphs of $G'$ such that each subgraph is isomorphic to a graph in $\mathcal{P}$ and such that the graph obtained by removing $\mathcal{C}$ has no component of size more than $r$. This can be accomplished by first choosing $k$ subsets of vertices of $V(G')$, each of size at most $p$, and then checking that each subset induces a graph that contains a graph $P \in \mathcal{P}$ as a subset, where each vertex in the subset is in $P$. More formally, we choose $k$ subsets $S_1, \ldots, S_k$ of vertices of $V(G')$ of size at most $p$; there are $O(m^p)$ choices for each subset, and $O(m^{pk})$ choices for the set of $k$ subsets, where $m = |V(G')|$ (for the matching problem, we ensure that the subsets are disjoint). Checking if a particular graph $P \in \mathcal{P}$ is a subgraph of the graph induced on a set $S_i$ can be accomplished by trying all possible mappings of vertices to vertices and then checking for all edges, or $O(p^2 p!)$ time in total. Finally, to check that no component is of size greater than $r$, we use a linear number of steps. In total, the number of steps will thus be $O(m^{pk}(m+|\mathcal{P}|p^2p!)) = O(m^{pk}(m + |\mathcal{P}|(p + 3)!))$. We observe that since $G'$ contains the central set and at most $pk + 1$ satellites in each equivalence class, since there are $2^d = 2^{pkr+\binom{r}{2}}$ equivalence classes and each satellite has at most $r$ vertices, the size of $m$ is at most $pk+r(pk+1)2^{pkr+\binom{r}{2}} = O(r(pk+1)2^{r(pk+r)})$. Thus Step 4 can be executed in time $O(r^{pk}(pk + 1)^{pk}2^{pkr(pk+r)}(r(pk + 1)2^{r(pk+r)} + |\mathcal{P}|(p + 3)!))$.

Finally, we determine the running time of Step 5. By the argument in the previous paragraph, the number of possible solutions in $\mathbf{MCG}_k(G')$ is in $O(m^{pk}) = O(r^{pk}(pk+1)^{pk}2^{pkr(pk+r)})$. The size of $\mathbf{S}^*$ is no greater than the number of possible equivalence classes, which was shown in Lemma 3 to be in $O(2^{r(pk+r)})$.

There are $O(2^k)$ subsets of $\mathcal{G}$, as $|\mathcal{G}| \leq k$, and at most $k$ choices of $H$ for the same reason. The cost of computing the binomial coefficient is $pk + 1 = |\mathcal{R}_{\mathcal{S}}|$, assuming constant-time arithmetic operations. The total cost of Step 5 will thus be $O(r^{pk}(pk+1)^{pk}2^{pkr(pk+r)}2^{r(pk+r)}2^k k(pk+1))$.

The dominating steps are Steps 3 and 4; using the fact that $|\mathcal{P}| \leq 2^{\binom{p}{2}} \leq 2^{p^2}$, we obtain the claimed time bound of

$$O(n \cdot r(pk+r) + r^{pk}(pk+1)^{pk}2^{pkr(pk+r)}(r(pk+1)2^{r(pk+r)} + 2^{p^2} \cdot (p+3)!)).$$

We are able to count matchings by making a small modification in the algorithm of the previous theorem.

**Theorem 2.** *The value* $\mathsf{mmm}_k(G)$ *can be determined in time*

$$O(n \cdot r(pk+r) + r^{pk}(pk+1)^{pk}2^{pkr(pk+r)}(r(pk+1)2^{r(pk+r)} + 2^{p^2} \cdot (p+3)!)).$$

*Proof.* The proof of the theorem follows from the proof of Theorem 1 and the fact that we can compute $\mathsf{mmm}_k(G)$ by replacing the last two lines of the algorithm **COMPUTE-mcg-k**$(p, r, G)$ with the following lines.

```
COMPUTE-mmm_k$(p, r, G)$
4. Compute $\mathbf{MMM}_k(G')$ using brute force.
5. Compute and return the following number:
   $$\sum_{\mathcal{G} \in \mathbf{MCG}_k(G')} \prod_{\mathcal{S} \in \mathbf{S}^*} \sum_{\mathcal{G}' \subseteq \mathcal{G}} \binom{o_{\mathcal{S}} + |\{J \in \mathcal{R}_{\mathcal{S}} : V(J) \cap \mathbf{V}(\mathcal{G}') \neq \emptyset\}|}{|\{J \in \mathcal{R}_{\mathcal{S}} : V(J) \cap \mathbf{V}(\mathcal{G}') \neq \emptyset\}|}.$$
```

Here we cannot replace each graph in the solution independently; instead we choose a subset $\mathcal{G}'$ to replace and then consider all ways of choosing the right number of satellites in each class either from the remaining or the omitted satellites. The analysis is very similar to that given in the proof of Theorem 1.

## 6 Conclusions

Our primary concern in developing the algorithms was to maintain an additive, rather than multiplicative, relationship between the polynomial (on $n$) and non-polynomial (on the parameters) parts of the running time. We also stress that our analysis for determining the super-polynomial part of the time complexity is a worst-case analysis, and the algorithm should be expected to run even faster in practice, at least for small values of $k$.

We observe that the algorithm **COMPUTE-mcg-k**$(p, r, G)$ can be used to determine the existence of a $(k, r)$-central set, as the problems are equivalent for $p = 1$. We can thus count the number of $(k, r)$-central sets in a graph in time $O(n \cdot r(k + r) + 2^{O(kr(k+r))})$.

We leave as an open problem the question of whether enumerating all optimal solutions of the covering and matching problems is fixed-parameter enumerable.

## References

[AR02] V. Arvind and V. Raman. Approximation algorithms for some parameterized counting problems. In *Electronic Colloquium on Computational Complexity*, 2002.

[Dam04] P. Damaschke. Parameterized enumeration, transversals, and imperfect phylogeny reconstruction. In *Proc., 1st International Workshop on Parameterized and Exact Computation (IWPEC 2004)*, pages 1–12, 2004.

[DST04a] J. Díaz, M. Serna, and D. M. Thilikos. Fixed parameter algorithms for counting and deciding bounded restrictive list h-colorings. In *Proc., 12th Annual European Symposium on Algorithms (ESA 2004)*, pages 275–286, 2004.

[DST04b] J. Díaz, M. Serna, and D. M. Thilikos. Recent results on parameterized $h$-coloring. In Jarik Nesetril and P. Winkler (eds.) DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, Morphisms and Statistical Physics*, volume 63, pages 65–86. Amer. Math. Soc., Providence, RI, 2004.

[Fer02] H. Fernau. On parameterized enumeration. In *Proc., 8th Annual International Conference on Computing and Combinatorics (COCOON 2002)*, pages 564–573, 2002.

[FG04] J. Flum and M. Grohe. The parameterized complexity of counting problems. *SIAM J. Comput.*, 33(4):892–922 (electronic), 2004.