The Parameterized Complexity of Some Minimum Label Problems[☆]

Michael R. Fellows[a,1], Jiong Guo[b,2], Iyad Kanj[c,3]

[a]*The University of Newcastle, Callaghan, NSW 2308, Australia*
[b]*Universität des Saarlandes, Room 320, E 1.4, D-66123 Saarbrücken, Germany*
[c]*DePaul University, Chicago, IL 60604, USA*

Abstract

We study the parameterized complexity of several minimum label graph problems, in which we are given an undirected graph whose edges are labeled, and a property Π, and we are asked to find a subset of edges satisfying property Π with respect to G that uses the minimum number of labels. These problems have a lot of applications in networking. We show that all the problems under consideration are W[2]-hard when parameterized by the number of used labels, and that they remain W[2]-hard even on graphs whose pathwidth is bounded above by a small constant. On the positive side, we prove that most of these problems are FPT when parameterized by the solution size, that is, the size of the sought edge set. For example, we show that computing a maximum matching or an edge dominating set that uses the minimum number of labels, is FPT when parameterized by the solution size. Proving that some of these problems are FPT requires interesting algorithmic methods that we develop in this paper.

Key words: edge labeled graphs, W-hardness, fixed-parameter tractability
2000 MSC: 68R10, 05C85, 68Q17

1. Introduction

In this paper we consider several *minimum label graph problems* that are defined as follows:

> **Input**: A graph $G = (V, E)$ whose edges are associated with labels or colors specified by a function $C : E \to C$, where C denotes the set of labels (also referred to as colors in this paper), a graph property Π, and an integer d.
> **Output**: A set $E' \subseteq E$ such that the subgraph of G consisting of the set of edges in E' satisfies Π with respect to G, and the number of labels/colors used by the edges in E' is at most d.[4]

☆ A preliminary version of this paper appeared in WG 2009: 88-99.

Email addresses: michael.fellows@newcastle.edu.au (Michael R. Fellows), jguo@mmci.uni-saarland.de (Jiong Guo), ikanj@cs.depaul.edu (Iyad Kanj)

[1] Supported in this work by the Australian Research Council, and by a Research Award from the Alexander von Humboldt Foundation, Bonn, Germany.

[2] Supported by the Excellence Cluster on Multimodal Computing and Interaction (MMCI).

[3] This work was supported in part by a DePaul University Competitive Research Grant.

[4] The property Π may not only depend on the set of edges E', but rather on the set of edges E' and the graph G (e.g., the property of being a Hamiltonian cycle of G).

Minimum label problems have been extensively studied in the last few years. These problems are motivated by applications from telecommunication networks, electrical networks, and multi-modal transportation networks. For example, in communication networks, there are different types of communication media, such as optic fiber, cable, microwave, and telephone line. A communication node may communicate with different nodes by choosing different types of communication media. Given a set of communication network nodes, the problem of finding a connected communication network using as few types of communication media (i.e., labels/colors) as possible is exactly the Minimum Label Spanning Tree problem, in which the property Π is the property of being a spanning tree of G (see [6, 20] for more details). Among the minimum label problems that have been extensively studied, we mention the Minimum Label Spanning Tree problem [2, 3, 4, 6, 11, 15, 20, 21, 24, 25, 26], the Minimum Label Path problem [3, 5, 11, 23, 27] (where Π is the property of being a path between two designated vertices), the Minimum Label Cut problem [13, 27] (where Π is the property of being a cut between two designated vertices), and the Minimum Label Perfect Matching problem [16] (where Π is the property of being a perfect matching).

The previous work on minimum label problems mainly dealt with determining the classical complexity of these problems and studying their approximability. Some of the previous work, however, dealt with developing exact algorithms for these problems. For example, Broersma et al. [3] devised two exact algorithms for the Minimum Label Path and Minimum Label Cut problems with running time $O(n \cdot \min\{|C|^{d(s,t)}, 2^{|C|}\})$ and $O(n^2 \cdot |C|!)$, respectively, where C denotes the set of labels (colors), and $d(s,t)$ denotes the distance between the two designated vertices s and t.

In the current paper we study the parameterized complexity of several minimum label graph problems, with respect to two natural parameters: the number of used labels d, and the size of the solution $|E'|$. The problems under consideration are: Minimum Label Spanning Tree (MLST), Minimum Label Hamiltonian Cycle (MLHC) (where Π is the property of being a Hamiltonian cycle), Minimum Label Cut (MLC), Minimum Label Edge Domination Set (MLEDS) (where Π is the property of being an edge dominating set, that is, every edges in $E \setminus E'$ shares at least one endpoint with some edge in E'), Minimum Label Perfect Matching (MLPM), Minimum Label Maximum Matching (MLMM) (where Π is the property of being a maximum matching of G), and Minimum Label Path (MLP).

From some of the NP-hardness reductions for the above problems, we can derive parameterized intractability results with respect to the parameter d; for example, the NP-hardness reduction for Minimum Label Spanning Tree shows that this problem is W[2]-hard [15]. In this paper, we strengthen these intractability results by showing that, even on graphs whose pathwidth is at most a small constant, when parameterized by the number of used labels d, these problems remain W[2]-hard. These results are interesting, as very few natural parameterized problems are known to be (parameterized) intractable on graphs with bounded pathwidth. When parameterized by the solution size $|E'|$, we show that, with the only exceptions of Minimum Label Path and Minimum Label Cut, which we prove to be W[1]-hard, all other problems are fixed-parameter tractable (on general graphs). Showing that some of these problems are FPT is non-trivial, and requires interesting algorithmic methods that we develop in this paper.

We start by giving the necessary background and terminology in Section 2. All the hardness results will be presented in Section 3, while Section 4 contains all the fixed-parameter tractability results. Finally, we give some concluding remarks in Section 5.

2. Preliminaries

Throughout this paper we only consider finite undirected graphs that are simple (i.e., with no loops or multiple edges). Our terminology and definitions generally agree with West [22].

For a graph G, we denote by $V(G)$ and $E(G)$ the set of vertices and edges of G, respectively, and by $n(G)$ and $e(G)$ the number of vertices and edges in G, respectively. For a vertex v, we denote by $N(v)$ the set of neighbors of v. The *degree* of a vertex v in G is $|N(v)|$. We shall denote the degree of a vertex v in G by $\deg(v)$, and its degree in a subgraph $H \subseteq G$ by $\deg_H(v)$. For a vertex v in $V(G)$, we denote by $G - v$ the graph obtained from G by removing v and its incident edges, and by $G - e$, the graph obtained from G by removing the edge e while keeping its endpoints. For a subset of vertices (resp. edges) S in G, we denote by $G[S]$ the subgraph of G induced by S (resp. induced by the endpoints of the edges in S). The *size* of S is its cardinality.

A *matching* in a graph G is a set of edges M such that no two edges in M share the same endpoint. A matching M is said to be *maximum* if M has the maximum size among all matchings in G. A matching M in G is *maximal* if $M \cup \{e\}$ is not a matching for every $e \in E(G) \setminus M$.

A set of edges S in G is said to be an *edge-dominating set* for G if for every edge e in $E(G) \setminus S$, e is incident on at least one edge in S.

A *parameterized problem* is a set of instances of the form (x, k), where $x \in \Sigma^*$, for some finite alphabet Σ, and k is a non-negative integer called the *parameter*. A parameterized problem Q is *fixed-parameter tractable*, or simply FPT, if there exists an algorithm A that on input (x, k) decides if (x, k) is a yes-instance of Q in time $f(k)|x|^{O(1)}$, where f is a recursive function independent of $|x|$. In analogy to the polynomial time hierarchy, a hierarchy for parameterized complexity, called the *W-hierarchy*, has been defined. At the 0th level of this hierarchy lies the class of fixed-parameter tractable problems FPT. The class of all problems at the ith level of the W-hierarchy ($i > 0$) is denoted by *W[i]*. A parameterized-complexity preserving reduction (FPT-reduction) has been defined as follows. A parameterized problem Q is *FPT-reducible* to a parameterized problem Q' if there exists an algorithm of running time $f(k)|x|^{O(1)}$ that on an instance (x, k) of Q produces an instance $(x', g(k))$ of Q' such that (x, k) is a yes-instance of Q if and only if $(x', g(k))$ is a yes-instance of Q', where the functions f and g depend only on k. A parameterized problem Q is *W[i]-hard* if every problem in W[i] is FPT-reducible to Q. Many well-known problems have been proved to be W[1]-hard including: CLIQUE and INDEPENDENT SET. Examples of W[2]-hard problems include SET PACKING, DOMINATING SET, HITTING SET and SET COVER. The *parameterized complexity hypothesis*, which is a working hypothesis for parameterized complexity theory, states that W[i] $\neq$ FPT for every $i > 0$. The reader is referred to Downey and Fellows' book [8] for more details about parameterized complexity theory.

3. Parameterized Hardness Results

First, we show that even on graphs whose pathwidth is at most a small constant, all the considered minimum label problems are W[2]-hard, when parameterized by the number of used labels d. These results are very interesting since few problems are known to be W-hard on graphs of bounded pathwidth. For more details on pathwidth, we refer the reader to [14].

Theorem 3.1. *Parameterized by the number of used labels d:*

- MINIMUM LABEL EDGE DOMINATING SET *(MLEDS) and* MINIMUM LABEL MAXIMUM MATCHING *(MLMM) are W[2]-hard on trees of pathwidth at most 1;*

- Minimum Label Spanning Tree *(MLST) and* Minimum Label Path *(MLP) are W[2]-hard on graphs with pathwidth at most 2;*

- Minimum Label Cut *(MLC) and* Minimum Label Perfect Matching *(MLPM) are W[2]-hard on graphs with pathwidth at most 3; and,*

- Minimum Label Hamiltonian Cycle *(MLHC) is W[2]-hard on graphs with pathwidth at most 5.*

Proof. All the corresponding FPT-reductions are from the W[2]-hard Hitting Set (HS) problem [8], defined as follows. Given a ground set S, a collection $\mathcal{L}$ of subsets of S, and a nonnegative integer k, decide if there exists a subset S' of S of cardinality at most k, such that every subset in $\mathcal{L}$ has a non-empty intersection with S'. We assume that $\mathcal{L} = \{c_1, c_2, \ldots, c_m\}$.

To show the hardness of MLEDS and MLMM, we construct for every subset c_i in $\mathcal{L}$ a star with $|c_i|$ many leaves. The edges between the root vertex of the star and its leaves are labeled with the elements of c_i. Then we add another $m-1$ vertices $r_1, r_2, \ldots, r_{m-1}$ and connect the root vertices of the stars for c_i and c_{i+1} to r_i, for $1 \leq i \leq m-1$. All edges incident to the r_i's are labeled with distinct labels that are not in S. The resulting graph T is clearly a caterpillar whose minimum edge dominating sets and maximum matchings have size $|\mathcal{L}|$. It is well-known that caterpillars have pathwidth 1. The edges labeled by the elements of every size-k hitting set of $\mathcal{L}$ dominate all edges of T and form a maximum matching. It is also not hard to see that there exist a minimum edge domination set and a maximum matching of T which do not contain any edge incident to the r_i's. This gives the correctness of the reduction.

Next, consider MLST. As in the MLEDS and MLMM cases, for each subset c_i in $\mathcal{L}$, we add a star consisting of a root vertex and $|c_i|$ leaves. The edges in this star are labeled with the elements of c_i. Then, we connect the leaves of this star by a path[5] whose edges have the same label x, where $x \notin S$. Finally, we connect all the root vertices of the stars by a path whose edges have the same label x. Clearly, the resulting graph has pathwidth 2, since we can construct a path decomposition where for a subset $c_i \in \mathcal{L}$ there are $|c_i| - 1$ bags, each of which contains the root vertex of the star corresponding to c_i and two leaves of this star that became adjacent after connecting the leaves of this star by a path. Observe that every size-k solution of the HS-instance corresponds to a solution of the resulting MLST-instance using $k+1$ labels, and vice versa. This gives the W[2]-hardness of MLST.

For MLP, we first add $m+1$ vertices $r_0, r_1, \ldots, r_m$. Then, for each $c_i \in \mathcal{L}$, we add $|c_i|$ many degree-2 vertices which are common neighbors of r_{i-1} and r_i. This means that there are $|c_i|$ many edge-disjoint length-2 paths between r_{i-1} and r_i, where the two edges of each path are labeled with a distinct element of c_i. Finally, let $s := r_0$ and $t := r_m$. The created graph has pathwidth two, since for each of the subgraphs induced by r_{i-1}, r_i, and the vertices corresponding to c_i, $1 \leq i \leq m-1$, we can create a path decomposition with $|c_i|$ bags, each of which contains r_{i-1}, r_i, and one of the $|c_i|$ vertices corresponding to c_i. Every size-k hitting set gives a path of length $2|\mathcal{L}|$ between s and t with k labels.

The reduction for MLC consists of $|\mathcal{L}|$ paths between two designated vertices s and t. These paths are vertex-disjoint, with the only exception of s and t; each path represents a subset c_i of $\mathcal{L}$ and its edges are labeled (in a one-to-one fashion) by the elements in c_i. Observe that without s and t the constructed graph consists of disjoint paths whose pathwidth is 1. Adding s and t to

[5]By connecting vertices $w_1, \ldots, w_\ell$ by a path we mean adding the edges $\{w_j, w_{j+1}\}$, for $j = 1, \ldots, \ell - 1$.

all bags of the corresponding width-1 path decomposition shows that the pathwidth of the whole graph is at most 3. To cut all these paths, one needs to delete exactly $|\mathcal{L}|$ edges, whose labels correspond then to a hitting set of $\mathcal{L}$.

In the graph constructed for MLPM, for each c_i in $\mathcal{L}$, we create two copies of a star consisting of a root vertex and $|c_i|$ leaves. The edges in each copy are labeled with the elements of c_i. Then, for each $c_i \in \mathcal{L}$, we connect the two copies of the star for c_i by adding an edge between every two leaves (one from each copy) corresponding to the same element in c_i. All edges between the two copies are labeled by the same label $x \notin S$. Since deleting the two root vertices of the two copies of the star for c_i results in a vertex-disjoint union of edges, the whole graph has pathwidth at most 3. Clearly, every perfect matching of the resulting graph contains $\sum_{c_i \in \mathcal{L}}(|c_i| - 1)$ edges labeled by x, and $2|\mathcal{L}|$ edges from the stars. The labels of the $2|\mathcal{L}|$ edges give then a hitting set of $\mathcal{L}$.

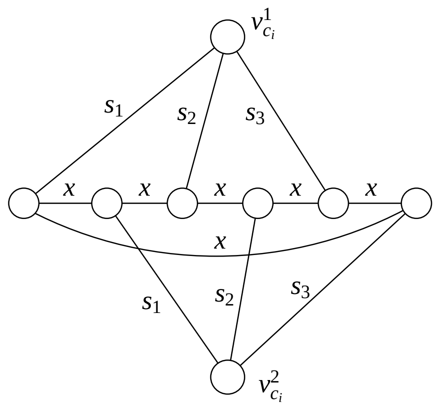

Figure 1: The gadget for a subset $c_i = \{s_1, s_2, s_3\} \in \mathcal{L}$ used in the reduction for MLHC. Note that $x \notin S$.

Finally, we present the reduction for MLHC. We add a gadget for every subset $c_i \in \mathcal{L}$, as shown in Figure 1. Then, we connect $v^2_{c_i}$ with $v^1_{c_{i+1}}$, for all $1 \le i \le m-1$, and $v^2_{c_m}$ with $v^1_{c_1}$. These $|\mathcal{L}|$ edges are labeled by $x \notin S$. This graph has pathwidth at most 5, since each gadget shown in Fig. 1 has clearly a pathwidth of at most 4, and adding $v^1_{c_1}$ to all bags of the decompositions of these gadgets gives a path decomposition of the whole graph with pathwidth 5. Every Hamiltonian cycle enters or leaves the gadget for c_i at $v^1_{c_i}$ or $v^2_{c_i}$. The only possibility to go through all vertices in the middle involves passing through an edge of label x. It is easy to verify that $\mathcal{L}$ has a hitting set of size at most k if and only if there is a Hamiltonian cycle in the resulting graph that uses at most $k + 1$ labels. □

Next, we consider Minimum Label Cut (MLC) and Minimum Label Path (MLP) with the size of the set E' as the parameter.

Theorem 3.2. *Parameterized by the solution size $|E'|$:*

- Minimum Label Cut *is W[1]-hard on graphs with pathwidth at most 4, and*
- Minimum Label Path *is W[1]-hard on graphs with pathwidth at most 2.*

Proof. We give two FPT-reductions from the W[1]-hard Multicolored Clique problem [9]. Multicolored Clique has as input a graph G, together with a proper k-coloring of the vertices of G, and the question is whether there is a k-clique in G consisting of exactly one vertex from each color class. The parameter is the clique size k.

To construct an MLC-instance from a MULTICOLORED CLIQUE instance $(G = (V, E), k)$, we partition E into $\binom{k}{2}$ subsets, each containing the edges between two color classes. For each subset of edges, we create in the MLC-instance a path between two designated vertices s and t whose length is equal to the size of this subset; each edge of the path is in a one-to-one correspondence with an edge in this subset. Finally, we replace each edge of the path by two parallel length-2 paths, and these two length-2 paths are labeled by the two endpoints of the corresponding edge in E, respectively; that is, each length-2 path is labeled by an endpoint of the edge. In the resulting MLC-instance we ask for an s-t cut of size at most $2 \times \binom{k}{2}$, using at most k labels.

Since there are exactly $2 \times \binom{k}{2}$ edge-disjoint paths between s and t, every solution of the MLC-instance contains exactly $2 \times \binom{k}{2}$ edges whose labels correspond to k vertices from the MULTICOLORED CLIQUE instance. Those vertices must induce exactly $\binom{k}{2}$ many edges in G. The converse is also easy to check. Thus, there is a correspondence between the solutions of both instances. Moreover, since the subgraph corresponding to a subset of the edge partition has clearly pathwidth 2, and adding s and t to all bags of the path decompositions of these subgraphs gives a path decomposition of the whole graph, the resulting MLC-instance is clearly a graph whose pathwidth is equal to 4.

The FPT-reduction for MINIMUM LABEL PATH works analogously. Here, we introduce first $l := \binom{k}{2} + 1$ many vertices $r_1, r_2, \ldots, r_l$. Then, as in the MLC-case, we partition the set of edges of G into $\binom{k}{2}$ subsets and add a gadget for the first subset between r_1 and r_2 and for the second subset between r_2 and r_3 and so on. For a subset with j many edges, the corresponding gadget consists of j many length-2 paths between the two corresponding r-vertices; each path represents an edge in this subset and thus its edges are labeled by the two endpoints of this edge. Finally, we set $s := r_1$ and $t := r_l$. Since the gadget between r_{i-1} and r_i has pathwidth 2 and the constructed MLP-instance consists of a linear ordering of such gadgets, the pathwidth of the MLP-instance is 2. Clearly, every path from s to t has length $2 \cdot \binom{k}{2}$, and, to construct such a path, we have to connect r_i with r_{i+1} by a length-2 path for every $1 \leq i \leq m - 1$. The labels of these length-2 paths represent a clique of the MULTICOLORED CLIQUE instance. $\square$

4. Fixed-Parameter Tractability Results

Parameterized by the solution size, MINIMUM LABEL SPANNING TREE, MINIMUM LABEL PERFECT MATCHING, and MINIMUM LABEL HAMILTONIAN CYCLE are all fixed-parameter tractable, since the instance size is bounded by a function of the parameter. However, it requires much more effort to show that MINIMUM LABEL MAXIMUM MATCHING (MLMM) and MINIMUM LABEL EDGE DOMINATING SET (MLEDS) are fixed-parameter tractable with respect to the same parameter.

4.1. Minimum Label Maximum Matching (MLMM)

We start by recalling the definition of MLMM:

Given: an undirected graph G, and a function C assigning each edge in $E(G)$ a label/color in $\{c_1, \ldots, c_p\}$
Output: a maximum matching M such that the number of labels/colors used by the edges in M is minimum, among all maximum matchings in G

Parameter: the size of a maximum matching in G

Let (G, k) be an instance of MLMM, where k is the size of a maximum matching in G. Let M be a maximal matching in G, $I = V(G) \setminus V(M)$, and note that I is an independent set in G. Recall that $G[M]$ denotes the subgraph of G induced by the endpoints of the edges in M.

The algorithm is a search-tree based algorithm: it starts by growing a set of partial solutions, i.e., matchings, into an optimal solution, i.e., a maximum matching that uses the minimum number of colors. To do so, the algorithm branches on some vertices and edges in G to decide whether they belong to an optimal solution or not. Since the branching will consider all possibilities, we will maintain the invariant that at least one partial solution, among all partial solutions we keep, can be extended to an optimal solution. The algorithm can be split into several stages, each trying to simplify the resulting instance further by possibly performing more branchings. In order for the reader to get a feel of what these stages are trying to achieve, and how together they contribute to the final solution, we give an intuitive description of each stage first.

In Stage 1, we branch on the vertices and edges in $G[M]$ to determine which ones belong to an optimal solution. At the end of this stage, the edges in $G[M]$ will be removed, as well as some of its vertices. We will be left with a bipartite graph whose first partite set S is a subset of vertices in $G[M]$, consisting of the endpoints of the edges that belong to an optimal solution (under the corresponding branching), and whose second partition is a subset of vertices in I. We note that during this stage some edges in $G[M]$ will be added to the partial solutions, and hence, their colors are decided to be used by the optimal solution. Moreover, the parameter k is decremented by a value equal to the number of edges added to the partial solution.

In Stage 2, we start with a bipartite graph $B = (S, I)$, and we would like to compute a maximum matching that matches S into I, and that uses the minimum number of colors, under the constraint that some colors have already been determined (from Stage 1) to be used by an optimal solution. In this stage we will simplify the instance further. We branch by enumerating all possible partitions of S into groups S_i, $i = 1, \ldots, \ell$, such that there is an optimal solution in which all vertices in S_i are matched using edges of the same color—we will call such a set of edges a *monochromatic matching*. For a fixed partitioning of S into groups, we compute, for each group S_i, the set $\mathcal{M}_i$ of monochromatic matchings that match S_i into I. If $|\mathcal{M}_i|$ is bounded above by a predefined function of k, then we can compute a matching in $\mathcal{M}_i$ that is part of an optimal solution by trying (branching on) all monochromatic matchings in $\mathcal{M}_i$, and subsequently remove S_i from S. If all monochromatic matchings in $\mathcal{M}_i$ use the same color, we branch on every vertex in $\mathcal{M}_i$ whose degree in $\mathcal{M}_i$ is larger than a predefined function of k.

In Stage 3, we can assume that, for each remaining group S_i in the resulting instance (S', I'), $|\mathcal{M}_i|$ is larger than a predefined function of the parameter, and for each $\mathcal{M}_i$ whose monochromatic matchings all use the same color, the degree of every vertex in $\mathcal{M}_i$ is larger than a predefined function of the parameter. We show in this case that an optimal solution can be computed easily (without any branching): a matching M' that matches S' into I' exists, such that the set of edges in M' incident on each group S_i is a monochromatic matching in $\mathcal{M}_i$.

We now describe these three stages in more detail.

Stage 1

Let M_{opt} be an optimal solution that we are trying to compute. We apply the algorithm **Stage-1-Algorithm** given in Figure 2. We note that if at any point in the algorithm **Stage-1-Algorithm** the partial solution contains two edges that share an endpoint, then the partial solution can be rejected. The case is similar if the partial solution's size exceeds the parameter k. We do not list these rejection scenarios in the algorithm in order to keep the description of the algorithm simple.

Algorithm **Stage-1-Algorithm**

1. For every edge e in $G[M]$ we branch as follows:
   1.1. **Case** e **in** M_{opt}: in this case we include e, decrement k by 1, and remove e and its endpoints from the graph; we also record that the color $C(e)$ is used in the optimal solution by adding it to a set of used colors C_{used};
   1.2. **Case** e **is not in** M_{opt}: in this case we simply remove e, that is, we set $G := G - e$;
2. For every remaining vertex v in $G[M]$ we branch as follows:
   2.1. **Case** v **is an endpoint of an edge in** M_{opt}: in this case we keep v in the graph;
   2.2. **Case** v **is not an endpoint of an edge in** M_{opt}: in this case we remove v by setting $G := G - v$;

Figure 2: The algorithm for Stage 1.

Note that the case distinction in **Stage-1-Algorithm** corresponds to the possible cases resulting from enumerating whether a vertex/edge is in the optimal solution M_{opt} or not, and hence this case distinction is done without knowing the optimal solution M_{opt}.

Proposition 4.1. *The branching performed in* **Stage-1-Algorithm** *is correct.*

Proof. First note that the algorithm considers all possibilities when branching on an edge or a vertex.

In step 1.1, when the edge e is included in the partial solution, none of its endpoints can any longer be used as an endpoint of any other edge in M_{opt}; this justifies the removal of the endpoints of e from the graph in this case. The situation is different in step 1.2: the endpoints of edge e can still be used as endpoints of some other edges in M_{opt}, and hence they must be kept in the graph.

In step 2.1, when a vertex has been decided to be an endpoint of some edge in M_{opt}, the vertex remains in the graph because the other endpoint of that edge has not been decided yet. In step 2.2, when the vertex has been decided not be an endpoint of an edge in M_{opt}, it is simply removed from the graph.

Therefore, the branching performed by the algorithm is exhaustive (covers all possibilities), and the corresponding steps taken are correct. ☐

Let S be the set of remaining vertices in $G[M]$, and note that since all the edges in $G[M]$ have been removed during the branching, S is an independent set. Moreover, under the working assumption that our partial solution (branching) is valid (i.e., leads to an optimal solution), every vertex in S must be an endpoint of an edge in the optimal solution M_{opt}. Therefore, the number of vertices in S is at most k.

Let $B = (S, I)$ be the resulting bipartite graph from G after the branching. Note that the size of S, plus the number of edges in the partial solution, should add up to k at this point; otherwise, we can reject the partial solution.

The remaining task amounts to computing a matching with the minimum number of colors that matches S into I, under the constraint that some of the colors—those which appear in C_{used}—have been used.

Analysis of the number of partial solutions enumerated in Stage 1

To analyze the number of partial solutions generated in Stage 1, we count the number of paths in the search tree corresponding to the branching performed by the algorithm **Stage-1-Algorithm**. We have the following proposition:

Proposition 4.2. *The number of paths in the search tree corresponding to* **Stage-1-Algorithm**, *and hence the number of partial solutions generated by* **Stage-1-Algorithm**, *is* $O((8ek)^k)$, *where* e *is the base of the natural logarithm.*

Proof. Since $|M| \leq k$, the number of vertices in $G[M]$ is at most $2k$, and the number of edges in $G[M]$ is at most $\binom{2k}{2} = k(2k-1)$.

The branching in Stage 1 can be implemented as follows. For each $i = 0, \ldots, k$, we choose a matching of size i from the edges in $G[M]$ to be included in M_{opt}. For each of the remaining at most $(2k-2i)$ vertices in $G[M]$, we branch on it as indicated above, thus creating at most 2^{2k-2i} partial solutions. Therefore, the number of partial solutions enumerated in Stage 1 is bounded above by:

$$\begin{aligned} \sum_{i=0}^{k} \binom{k(2k-1)}{i} 2^{2k-2i} &= 4^k \sum_{i=0}^{k} \binom{k(2k-1)}{i} 1/4^i && (1) \\ &\leq 4^k \binom{k(2k-1)}{k} \sum_{i=0}^{k} 1/4^i && (2) \\ &\leq 4^k \cdot (e(2k-1))^k \cdot O(1) && (3) \\ &\leq 4^k \cdot (2ek)^k \cdot O(1) = O((8ek)^k). \end{aligned}$$

Inequality (2) is justified by the fact that the coefficient $\binom{k(2k-1)}{k}$ is the largest coefficient in the summation. Inequality (3) uses the fact that $\binom{n}{k} \leq (en/k)^k$, where e is the base of the natural logarithm (for instance, see [7]). It follows that the number of partial solutions enumerated in Stage 1 is $O((8ek)^k)$. □

Stage 2

Given the bipartite graph $B = (S, I)$ and the parameter $k' = |S| \leq k$, we try in this stage to simplify the instance further by performing more branching. For this purpose, the following notion will be helpful:

Definition 4.3. A matching is *monochromatic* if all its edges have the same color. If M' is a monochromatic matching, we denote by $C(M')$ the color of the edges in M'.

We would like to partition S into groups such that all vertices in the same group are matched in M_{opt} by a monochromatic matching of a distinct color (from any other group). To do so, we will enumerate all possible partitions of S. For a fixed partition of S into ℓ groups $S_1, \ldots, S_\ell$, we work under the assumption that, in M_{opt}, the vertices in each group S_i are matched by a monochromatic matching of a distinct color (from the colors of the other groups). Clearly, there exists at least one partition of S for which this working hypothesis is true, namely the one induced by the color classes in M_{opt}.

Let $S_1, \ldots, S_\ell$ be a fixed partition of S into ℓ nonempty groups, where $1 \leq \ell \leq k'$ is an integer. It is possible that a group S_i uses the color of an edge that was added to a partial solution in Stage 1; that is, a color that appears in C_{used}. Therefore, for each (possibly empty) subset C' of C_{used}, we try all one-to-one mappings from C' to $\{S_1, \ldots, S_\ell\}$. Fix such a mapping. Then some groups in $\{S_1, \ldots, S_\ell\}$ have been assigned colors, and hence the colors of the monochromatic matchings sought for these groups are fixed. Clearly, under the assumption that our partition of the vertices of S is correct, and since we are trying all possible assignments from the used colors to the groups, there will be an assignment of colors that corresponds to that in M_{opt}, and hence we are safe.

Definition 4.4. Let S_i, $i \in \{1, \ldots, \ell\}$, be a group. If S_i has a preassigned color, let c_i be this color and define $\mathcal{M}_i = \{M_i \mid M_i$ is a monochromatic matching that matches S_i into I and $C(M_i) = c_i\}$. Otherwise, the color of S_i is undetermined yet, and in this case define $\mathcal{M}_i = \{M_i \mid M_i$ is a monochromatic matching that matches S_i into $I\}$.

Let $h(k')$ be a function of k' whose value will be determined in Lemma 4.12. Let $\mathcal{P} = \{S_1, \ldots, S_\ell\}$ be a fixed partition of S, as discussed above. We perform more branching to simplify the instance by applying the algorithm **Stage-2-Algorithm** given in Figure 3.

Algorithm **Stage-2-Algorithm**

1. If there exists a group $S_i \in \mathcal{P}$ such that $|\mathcal{M}_i| \leq h(k')$, then branch on every matching in $\mathcal{M}_i$ as the matching that matches S_i in M_{opt}; for each branch corresponding to a matching M_i in $\mathcal{M}_i$:

   1.1. add the edges in M_i to the potential solution and decrement k' by $|S_i|$;
   1.2. remove the vertices in $V(M_i)$ from the graph, and remove every edge whose color is $C(M_i)$ from the graph;
   1.3. apply **Stage-2-Algorithm** recursively after setting $\mathcal{P} := \mathcal{P} \setminus \{S_i\}$;

2. else if there exists a group $S_i \in \mathcal{P}$ such that the total number of different colors used by the monochromatic matchings in $\mathcal{M}_i$ is more than 1 but not more than $h(k')$, then branch by trying all possible colors appearing in $\mathcal{M}_i$ to determine the color used in M_{opt} to match S_i (this color has to be one of the colors used by a monochromatic matching in $\mathcal{M}_i$); for each color c used by a monochromatic matching in $\mathcal{M}_i$:

   2.1. remove every edge incident on some vertex in $V(S_i)$ whose color is different from c;
   2.2. remove all edges of color c that are not incident on vertices in $V(S_i)$ from the graph;
   2.3. apply **Stage-2-Algorithm** recursively;

3. else if there exists a group $S_i \in \mathcal{P}$ such that all the monochromatic matchings in $\mathcal{M}_i$ have the same color, and if there exists a vertex v in S_i such that the number of edges incident on it in the matchings in $\mathcal{M}_i$ is at most $h(k')$, then branch on which edge in a matching in $\mathcal{M}_i$ matches v in M_{opt}; for each branch corresponding to an edge e_v:

   3.1. add e_v to the potential solution, remove the endpoints of e_v from the graph, and decrement k' by 1;
   3.2. update S_i and apply **Stage-2-Algorithm** recursively;

Figure 3: The algorithm for Stage 2.

We note that if the set $\mathcal{M}_i$ is empty for some group S_i, then the partial solution can be rejected since this would imply that the enumerated partition $\mathcal{P}$, or the color assignment to the groups in $\mathcal{P}$ is not valid. Again, we do not mention the rejection scenarios in the algorithm in order to keep the presentation simple.

Proposition 4.5. *The branching performed in* **Stage-2-Algorithm** *is correct.*

Proof. Under the working hypothesis that the fixed partition $\mathcal{P}$ is correct, that is, corresponds to that in M_{opt}, the branchings performed in steps 1, 2, and 3 exhaust all possibilities, and hence are correct in the sense that at least one of the paths in the search tree corresponding to the algorithm **Stage-2-Algorithm** will lead to an optimal solution. We justify next the operations performed in each step of the algorithm.

In step 1.1, since the branch assumes that the vertices in S_i are matched by M_i in M_{opt}, the edges in M_i are added to the partial solution, and the parameter k' is decremented by the number of these edges, that is by $|M_i| = |S_i|$. Step 1.2 removes the vertices in $V(M_i)$ because none of them can serve as an endpoint of any other edge in M_{opt}. Note also that, under the working assumption that only the vertices in S_i are matched by edges of color $C(M_i)$, any edge in the graph that is not incident on $V(S_i)$ and whose color is $C(M_i)$, is not used by M_{opt}, and hence can be removed from the graph. Step 1.3 applies the algorithm recursively after removing the group S_i from $\mathcal{P}$, since the vertices in this group have been removed from the graph.

In step 2.1, since the branch assumes that the vertices in $V(S_i)$ are matched by edges whose color is c, any edge incident on a vertex in $V(S_i)$ whose color is different from c is not used by M_{opt}, and hence can be removed from the graph. By the same token, no vertex that is not in $V(S_i)$ can be matched by an edge of color c, and hence edges of color c whose both endpoints are not in $V(S_i)$ can be removed; this justifies step 2.2. Finally, step 2.3 applies the algorithm recursively.

In step 3.1, since edge e_v has been decided to be the edge used by M_{opt} to match vertex v, e_v is added to the partial solution, its endpoints are removed from the graph (since these endpoints can no longer be used as the endpoints of another edge in M_{opt}), and the parameter k' is decremented by 1, reflecting the addition of e_v to the partial solution. Finally, group S_i is updated by removing vertex v from it, and the algorithm is applied recursively.

We conclude that the branching done in the algorithm is exhaustive, and the corresponding steps taken are correct. □

Let k'' be the resulting parameter after the execution of **Stage-2-Algorithm** above. The following hold true:

Proposition 4.6. *For each remaining group S_i, $i \in \{1, \ldots, \ell\}$:*

(i) $|\mathcal{M}_i| > h(k'')$.

(ii) *Either the number of colors appearing in $\mathcal{M}_i$ is more than $h(k'')$, or it is exactly 1.*

(iii) *If $\mathcal{M}_i$ has exactly one color appearing in it, then every vertex in S_i has more than $h(k'')$ edges that are incident on it in the matchings in $\mathcal{M}_i$.*

Proof. Part (*i*) follows from the fact that, after the execution of the algorithm, there will be no group S_i satisfying the condition in step 1, and hence there will be no group S_i for which the number of monochromatic matchings in $\mathcal{M}_i$ is at most $h(k'')$.

Part (*ii*) follows from the fact that, after the execution of the algorithm, there will be no group S_i satisfying the condition in step 2.

Part (*iii*) follows from the fact that, after the execution of the algorithm, there will be no group S_i for which the number of colors used by the monochromatic matchings in $\mathcal{M}_i$ is 1, and in which there exists a vertex v whose number of incident edges in $\mathcal{M}_i$ is at most $h(k'')$. □

In the next stage we show how, given the above Proposition, we can easily compute a solution to the resulting instance.

Analysis of the number of partial solutions enumerated in Stage 2

Lemma 4.7. *The number of paths in the search-tree corresponding to the enumeration of the different partitions of S, and the different assignments of used colors to the groups in each partition, is at most $2^{c_{used}} k'^{k'+1}(k'!)$, where c_{used} is the number of colors in C_{used}.*

Proof. Let c_{used} be the number of colors in C_{used}. The number of partitions of S into ℓ groups is at most $\ell^{|S|} \leq \ell^{k'}$, where $k' = |S|$. For each partition of S into ℓ groups, and for each subset C' of C_{used}, where $\ell \geq |C'|$, we map the colors in C' in a one-to-one fashion to a subset of the ℓ groups. There are at most $\ell!/(\ell - |C'|)! \leq \ell!$ such mappings. Therefore, the total number of partitions of S in which some of the ℓ groups have been assigned colors is at most $2^{c_{used}} \sum_{\ell=1}^{k'} \ell^{k'} \ell! \leq 2^{c_{used}} k'^{k'+1}(k'!)$. □

Let $\mathcal{P} = \{S_1, \ldots, S_\ell\}$, where $\ell \leq k'$, be a fixed partition of S in which some of the groups in S (possibly) have preassigned colors.

Lemma 4.8. *For each group S_i, $i \in \{1, \ldots, \ell\}$, we can determine if $|\mathcal{M}_i| \leq h(k')$, and if so, compute the monochromatic matchings in $\mathcal{M}_i$, in time $O(e(G)\sqrt{n(G)} + n(G)h(k'))$. Therefore, we can determine if there exists a group S_i such that $|\mathcal{M}_i| \leq h(k')$ in time $O(k'e(G)\sqrt{n(G)} + k'h(k')n(G))$.*

Proof. Let S_i be a group. We compute at most $h(k') + 1$ monochromatic matchings $M_i \in \mathcal{M}_i$. At the end, either we manage to compute $h(k') + 1$ monochromatic matchings in $\mathcal{M}_i$, and hence we have determined that $|\mathcal{M}_i| > h(k')$, or we know that $|\mathcal{M}_i| \leq h(k')$.

To do so, we iterate over each color c, and compute monochromatic matchings of color c that match S_i into I until either the total number of monochromatic matchings computed so far is $h(k') + 1$, or there are no more monochromatic matchings of color c that match S_i into I; at that point we try the next color. (If S_i has a preassigned color, then there is no need to iterate over each color, and we only consider the color assigned to S_i.) For a fixed color c, we consider the subgraph of B consisting of the edges of color c incident on vertices in S_i. Note that each matching in this subgraph that matches S_i into I is a maximum matching. It was shown in [19] how, after computing a maximum matching in a bipartite graph, every other maximum matching can be computed in linear time in the number of vertices of the subgraph, per matching. Therefore, computing at most $h(k') + 1$ monochromatic matchings of color c that match S_i into I can be done in time $O(e(G)\sqrt{n(G)} + n(G)h(k'))$, where $O(e(G)\sqrt{n(G)})$ is the time needed to compute the first monochromatic maximum matching for S_i [7]. As a matter of fact, since whenever we fix a color c for a group S_i we only look at the edges of color c incident on the vertices in S_i, and since we totally compute at most $h(k') + 1$ monochromatic matchings that match S_i, computing at most $h(k') + 1$ monochromatic matchings (regardless of the color) that match S_i can be done in time $O(e(G)\sqrt{n(G)} + n(G)h(k'))$. Since there are at most k groups, computing the sets $\mathcal{M}_i$, $i = 1, \ldots, \ell$, can be done in time $O(k'e(G)\sqrt{n(G)} + k'h(k')n(G))$. □

Lemma 4.9. *The number of paths in the search tree corresponding to the algorithm* **Stage-2-Algorithm** *is at most $h(k')^{2k'}$.*

Proof. We branch in the algorithm **Stage-2-Algorithm** in steps 1, 2, and 3. Each time we branch, we branch into at most $h(k')$ ways, and we end up calling the algorithm recursively. Therefore, to prove the lemma, it suffices to show that the depth of the recursion in the algorithm is at most $2k'$.

Note first that, whenever we branch in step 1 or step 3, the parameter is decremented by at least 1. Therefore, the total number of times we branch in steps 1 and 3, and hence the total number of times we call the algorithm recursively from steps 1 and 3, is at most k'.

In step 2, we only branch if the number of colors in $\mathcal{M}_i$ is more than 1 but less than $h(k')$. After branching, group S_i is assigned a color, and hence will never be considered again in step 2. Since there are at most k' groups, the number of times we branch, and hence we call the algorithm recursively, in step 2 is at most k'.

It follows that the depth of the recursion in the algorithm **Stage-2-Algorithm** is at most $2k'$, and the lemma follows. $\square$

Lemma 4.10. *The number of paths in the search tree corresponding to Stage 2 is at most* $2^{c_{used}}k'^{k'+1}(k'!)h(k')^{2k'}$.

Proof. We branch in Stage 2: (1) when we partition S into ℓ groups and assign some of these groups colors from the set C_{used}, and (2) when we apply the algorithm **Stage-2-Algorithm**.

By Lemma 4.7, the number of paths in the search tree corresponding to the branching mentioned in (1) above is $2^{c_{used}}k'^{k'+1}(k'!)$. By Lemma 4.9, the number of path in the search tree corresponding to the branching described in (2) above is $h(k')^{2k'}$.

It follows that the total number of paths in the search tree corresponding to the branching in Stage 2 is at most $(2^{c_{used}}k'^{k'+1}k'!) \cdot (h(k')^{2k'}) = 2^{c_{used}}k'^{k'+1}(k'!)h(k')^{2k'}$. $\square$

Lemma 4.11. *The running time along each path in the search tree corresponding to Stage 2 is* $O(k'^2 e(G)\sqrt{n(G)} + k'^2 h(k')n(G))$.

Proof. The running time along each path in the search tree corresponding to Stage 2 is the running time incurred by the execution of the algorithm **Stage-2-Algorithm**. In each call to **Stage-2-Algorithm**, the running time during this call is dominated by the running time of step 1 in the algorithm. This is because the computation in step 2 reduces to computing the number of colors appearing in $\mathcal{M}_i$, which is obviously dominated by the running time needed to compute $\mathcal{M}_i$. The running time in step 3 reduces to the running time incurred in computing the degree of every vertex in S_i, which is again dominated by the running time needed to compute $\mathcal{M}_i$.

By Lemma 4.9, the running time needed to compute the $\mathcal{M}_i$'s is $O(k'e(G)\sqrt{n(G)} + k'h(k')n(G))$. Along each path in the search tree we need to compute the $\mathcal{M}_i$'s at most $2k'$ times because the depth of the recursion in the algorithm **Stage-2-Algorithm** is at most $2k'$. It follows that the running time along each path in the search tree corresponding to Stage 2 is $O(k'^2 e(G)\sqrt{n(G)} + k'^2 h(k')n(G))$. $\square$

Stage 3

Given the resulting instance $B' = (S', I')$ from Stage 2, and the parameter $k'' = |S'|$, such that S' is partitioned into $S_1, \ldots, S_\ell$, where each set $\mathcal{M}_i$ associated with S_i, for $i = 1, \ldots, \ell$, satisfies the statements of Proposition 4.6, we show how to compute a matching M' that matches S' into I', and such that the set of edges in M' incident on S_i is a monochromatic matching whose edges are edges from the matchings in $\mathcal{M}_i$. We note that, at this point, the number of edges in the partial solution corresponding to the instance $B' = (S', I')$, plus the parameter k'', should add up to k; otherwise, the partial solution can be rejected.

Lemma 4.12. *Let $h(k'') \geq k''^2 + k''$. Assuming that each $\mathcal{M}_i$, $i = 1, \ldots, \ell$, satisfies Proposition 4.6, there exists a matching M' that matches S' into I', such that the set of edges in M' incident on S_i, for $i = 1, \ldots, \ell$, is a monochromatic matching whose edges are edges from the matchings in $\mathcal{M}_i$.*

Proof. Starting with S_1, we pick a monochromatic matching $M_1 \in \mathcal{M}_1$ that matches S_1 into I'. Let $I_1 = V(M_1) \cap I'$. Inductively, assume that we have determined a monochromatic matching M_j, where $1 \leq j < \ell$, such that the edges in M_j are edges from the matchings in $\mathcal{M}_j$, and such that $I_j = M_j \cap I'$ is disjoint from $I_1 \cup \ldots \cup I_{j-1}$. We show how to determine a monochromatic matching M_{j+1} whose edges are edges from the matchings in $\mathcal{M}_{j+1}$, and such that $I_{j+1} = M_{j+1} \cap I'$ is disjoint from $I_1 \cup \ldots \cup I_j$. We distinguish two cases:

Case 1. $\mathcal{M}_{j+1}$ contains more than $h(k'')$ colors. Since $|S'| = k''$, each vertex in I' has degree at most k''. Since $|I_1 \cup \ldots \cup I_j| \leq |S'| \leq k''$, and since (by the previous statement) each vertex in I' has degree at most k'', the number of edges incident on the vertices in $I_1 \cup \ldots \cup I_j$ is at most k''^2. Since $\mathcal{M}_{j+1}$ contains more than $h(k'') \geq k''^2 + k''$ monochromatic matchings of distinct colors, the number of monochromatic matchings in $\mathcal{M}_{j+1}$ whose edges are incident on some vertex in $I_1 \cup \ldots \cup I_j$ is at most k''^2. Therefore, the fact that $h(k'') > k''^2$ guarantees the existence of a monochromatic matching $M_{j+1} \in \mathcal{M}_{j+1}$ whose set of endpoints in I' is disjoint from $I_1 \cup \ldots \cup I_j$. Consequently, we can choose I_{j+1} to be disjoint from $I_1 \cup \ldots \cup I_j$.

Case 2. $\mathcal{M}_{j+1}$ contains a single color. By Proposition 4.6-(iii), every vertex in S_{j+1} has more than $h(k'')$ edges incident on it in $\mathcal{M}_{j+1}$. As in **Case 1** above, the number of edges incident on $I_1 \cup \ldots \cup I_j$ is at most k''^2. Since $h(k'') \geq k''^2 + k''$, for every vertex in S_{j+1}, there are at least k'' edges incident on it in $\mathcal{M}_{j+1}$ such that none of them is incident on a vertex in $I_1 \cup \ldots \cup I_j$. Moreover, all these edges (for all $v \in S_{j+1}$) have the same color. By Hall's theorem [22] (note that $|S_{j+1}| \leq k''$), there is a matching M_{j+1} whose edges are edges from the matchings in $\mathcal{M}_{j+1}$, and such that I_{j+1} is disjoint from $I_1 \cup \ldots \cup I_j$. $\square$

Analysis of the running time of Stage 3

This stage involves no enumerations. We have the following theorem:

Lemma 4.13. *The matching M' described in Lemma 4.12 can be computed in time $O(k''^3)$, where $k'' = |S'|$.*

Proof. Since S' contains at most k'' groups, it suffices to show that computing the matching M_j for each group S_j, as described in Lemma 4.12, can be done in time $O(k''^2)$.

If $\mathcal{M}_j$ satisfies **Case 1**, then in time $O(k''^2)$ we can find a color c appearing in $\mathcal{M}_j$ satisfying that no edge of color c is incident on a vertex in $I_1 \cup \ldots \cup I_{j-1}$. This is true because there are at most k'' vertices in $I_1 \cup \ldots \cup I_{j-1}$, each of degree at most k''. So we can compute the colors of the edges incident on the vertices in $I_1 \cup \ldots \cup I_{j-1}$ in time $O(k''^2)$, and hence determine a color c in $\mathcal{M}_j$ such that no edge of color c is incident on any vertex in $I_1 \cup \ldots \cup I_{j-1}$, and we can also determine a matching M_j of color c matching S_j into $I \setminus (I_1 \cup \ldots \cup I_{j-1})$.

If $\mathcal{M}_j$ satisfies **Case 2**, then, in time $O(k''^2)$, we can compute, for every vertex in S_j, k'' edges incident on it, none of which is incident on a vertex in $I_1 \cup \ldots \cup I_{j-1}$. This is true because the number of edges incident on the vertices in $I_1 \cup \ldots \cup I_{j-1}$ is at most k''^2. Once we have determined for every vertex in S_j k'' such edges, the matching M_j can be computed incrementally: for a vertex $v \in S_j$ pick one edge from its k'' incident edges that is not incident on any vertex in

$I_1 \cup \ldots \cup I_{j-1}$, nor on any edge which was previously placed in M_j; the existence of such an edge is guaranteed by the fact that $|S_j| \leq k''$ and that each vertex in S_j has at least k'' incident edges that are not incident on any point in $I_1 \cup \ldots \cup I_{j-1}$.

We conclude that the matching M' can be computed in $O(k''^3)$ time. □

Putting all together

The correctness of the algorithm follows from Proposition 4.1, Proposition 4.5, and Lemma 4.12. For each path in the search tree corresponding to the algorithm, either we reject the instance, or we end up computing a maximum matching that uses a certain number of colors. The maximum matching we output at the end is a maximum matching with the minimum number of colors.

The running time of the algorithm is bounded by the number of paths in the search tree (i.e., the number of partial solutions enumerated), multiplied by the time spent along each path. The number of paths in the search tree is the product of the number of paths in the search tree corresponding to Stage 1, which is $O((8ek)^k)$, and the number of paths in the search tree corresponding to Stage 2, which is $2^{c_{used}} k'^{k'+1}(k'!)h(k')^{2k'}$. Since each color c in C_{used} implies the existence of an edge of color c which was added to the partial solution in Stage 1, and since the size of an optimal solution is k, we conclude that $c_{used} + k' \leq k$. This, together with the choice of the function $h(k') = k'^2 + k'$, gives $2^{c_{used}} k'^{k'+1}(k'!)h(k')^{2k'} \leq k'^{c_{used}+k'+1}(k'!)(k'^2 + k')^{2k'}$ (assuming $k' \geq 2$), which is $O(k^{k+1}(k!)k^{4k}) = O(k^{6k+1}/e^k)$ (using Stirling's approximation [7]). It follows that the total number of partial solutions enumerated by the algorithm is $O(8^k k^{7k+1})$.

Along each path in the search tree, we spend time proportional to the size of the graph in Stage 1, that is $O(e(G)+n(G))$, and we spend $O(k'^2 e(G)\sqrt{n(G)} + k'^2(k'^2+k')n(G)) = O(k^2 e(G)\sqrt{n(G)} + k^4 n(G))$ time in Stage 2, and $O(k^3)$ time in Stage 3. It follows that the running time along each path in the search tree is $O(k^2 e(G)\sqrt{n(G)} + k^4 n(G))$. Consequently, the running time of the whole algorithm is $O(8^k k^{7k+5} e(G)\sqrt{n(G)})$.

Theorem 4.14. Minimum Label Maximum Matching *can be solved in time $O(8^k k^{7k+5} e(G)\sqrt{n(G)})$, and hence is FPT when parameterized by the size of the maximum matching in the graph.*

4.2. Minimum Label Edge Dominating Set (MLEDS)

Recall the definition of MLEDS:

Given: an undirected graph G, and a function C assigning each edge in $E(G)$ a label/color in $\{c_1, \ldots, c_p\}$
Output: an edge dominating set Q_{opt} of G of size at most k such that the number of labels/colors used by the edges in Q_{opt} is minimum
Parameter: k

The ideas used by the algorithm are similar in flavor to those used for the MLMM problem. Therefore, we will omit some details to avoid repetition. We start with the following easy observation:

Observation 4.15. *Let M be a matching in G, and let Q be an edge dominating set of G. Then $|Q| \geq |M|/2$.*

Let (G, k) be an instance of MLEDS. Let M be a maximal matching in G, $I = V(G) \setminus V(M)$, and note that I is an independent set in G. If $|M| > 2k$, then by Observation 4.15, G does not have an edge dominating set of size at most k, and we can reject the instance (G, k). Therefore, we may assume henceforth that $|M| \leq 2k$.

Similar to what we did for the MLMM problem, we will branch on the edges and vertices in $G[M]$ to determine which ones contribute to an optimal solution Q_{opt}, which is an edge dominating set of G of size at most k that uses the minimum number of colors (if such a solution exists). In the first stage, we apply the algorithm **Algo-I** given in Figure 4.

Algorithm **Algo-I**

1. For every edge $e \in G[M]$ we branch as follows:
   1.1. **Case** e **in** Q_{opt}: in this case we include e, decrement k by 1, set $G := G - e$, mark all edges incident on e as dominated, label both endpoints of e with the label "IN_{used}", and add the color $C(e)$ to C_{used};
   1.2. **Case** e **is not in** Q_{opt}: set $G := G - e$;
2. For every vertex v in $G[M]$ that is not labeled with IN_{used}, we branch as follows:
   2.1. **Case** v **is an endpoint of an edge in** Q_{opt}: label v with the label "IN" and mark every edge incident on v as dominated;
   2.2. **Case** v **is not an endpoint of an edge in** Q_{opt}: label v with the label "OUT";

Figure 4: Branching on the vertices and edges in $G[M]$.

Note that the label IN_{used} is used to indicate that a vertex is an endpoint of some edge that is already decided to be in Q_{opt}, whereas the label IN is sued to indicate that a vertex is decided to be in Q_{opt} but has no incident edge that was decided to be in Q_{opt} yet. The label OUT is used to indicate that a vertex is decided not be an endpoint of an edge in Q_{opt}.

Note that since I is an independent set in G, every edge in G must be dominated by an edge in Q_{opt} having at least one endpoint in $G[M]$. In particular, this is true for every edge in $G[M]$. Therefore, after branching on the edges and vertices in $G[M]$, we need to check that, for every edge $e \in G[M]$ that was decided not to be in Q_{opt}, and subsequently removed from G, at least one of its endpoints has label IN or IN_{used}. If this is not the case, then the partial solution that we have enumerated is not valid, and we reject it.

After the above branching, all the edges of $G[M]$ are removed from G, and we end up with a bipartite graph $B = (S, I)$, where S consists of the set of remaining vertices in $G[M]$. Note that at this point the number of edges in the partial solution, plus the number of vertices of label IN, must be at most k; otherwise, we reject the instance.

Every vertex in S has one of the following labels: (1) IN_{used} indicating that the vertex is an endpoint of a known edge which was determined to be in Q_{opt}, (2) IN indicating that the vertex is the endpoint of some edge in Q_{opt} but this edge has not been determined yet, and (3) OUT indicating that the vertex is not an endpoint of an edge in Q_{opt}. The edges in B have one of two possible types: (1) dominated, those are the edges with at least one endpoint of label IN_{used} or IN, and (2) not dominated, and those are the edges whose endpoint in S is of label OUT.

Since we are trying all possible branches for the edges and vertices in $G[M]$, there is at least one search path corresponding to the algorithm that will lead to an optimal solution.

Proposition 4.16. *The number of paths in the search tree corresponding to the algorithm* **Algo-I** *is at most* $(128ek)^k$.

Proof. The number of partial solutions enumerated by the branching can be upper bounded in a similar fashion to that in Stage 1 of the algorithm for MLMM. The only difference here is that the number of edges in the maximal matching M is at most $2k$, and hence, the number of vertices in $G[M]$ is at most $4k$, and consequently the number of edges in $G[M]$ is at most $2k(4k-1)$.

The branching can be implemented as follows. For each $i = 0, \ldots, k$, we choose a set of edges of size i from the edges in $G[M]$ to be included in Q_{opt}. For each of the remaining at most $(4k-2i)$ vertices in $G[M]$, we branch on it as indicated above, thus creating at most 2^{4k-2i} partial solutions. Therefore, the number of partial solutions enumerated is bounded above by:

$$\sum_{i=0}^{k} \binom{2k(4k-1)}{i} 2^{4k-2i} = 16^k \sum_{i=0}^{k} \binom{2k(4k-1)}{i} 1/4^i \tag{4}$$

$$\leq 16^k \binom{2k(4k-1)}{k} \sum_{i=0}^{k} 1/4^i \tag{5}$$

$$\leq 16^k \cdot (2e(4k-1))^k \cdot O(1) \tag{6}$$

$$= O((128ek)^k).$$

Inequality (5) is justified by the fact that the coefficient $\binom{2k(4k-1)}{k}$ is the largest coefficient in the summation. □

Now given the instance $B = (S, I)$, and the resulting parameter k', we will branch further to simplify the instance. First, observe that since the number of edges in Q_{opt} is at most k, the number of vertices in S that are labeled with IN_{used} or IN is at most $2k$; otherwise, we reject the partial solution.

Observation 4.17. *For every vertex w in I, the number of edges incident on w whose endpoint in S is labeled with IN_{used} or IN is at most $2k$.*

Let I_{in} be the set of vertices in I that are neighbors of vertices in S of label OUT. Then:

Proposition 4.18. $|I_{in}| \leq k$.

Proof. For every edge $e = \{u, v\}$ where $u \in S$ has label OUT, e needs to be dominated by an edge incident on v; therefore, the vertex v must be an endpoint of some edge in Q_{opt}. Since B is bipartite, for any two distinct vertices w_1 and w_2 in I_{in}, the set of edges between w_1 and its neighbors of label OUT in S, and the set of edges between w_2 and its neighbors of label OUT in S must be dominated by (at least) two distinct edges in Q_{opt}. Since the number of edges in Q_{opt} is at most k, there can be at most k vertices in I_{in} that are neighbors of vertices in S of label OUT. It follows that $|I_{in}| \leq k$. □

By Observation 4.17, every vertex in I has at most $2k$ edges incident on it whose endpoint in S is labeled IN_{used} or IN. Therefore, we will branch on every edge incident on a vertex in I_{in} whose endpoint in S is labeled IN_{used} or IN to determine if the edge is in Q_{opt} or not. We apply the algorithm **Algo-II** given in Figure 5.

It is easy to see that the branching performed in **Algo-II** is exhaustive and correct. After this branching, we check that for every vertex in I_{in}, at least one of the edges incident on it was decided to be in Q_{opt}; otherwise, we reject the partial solution.

Algorithm **Algo-II**

1. For every edge $e = \{u, v\}$ where $u \in S$ is labeled IN_{used} or IN and $v \in I_{in}$ we branch as follows:
   1.1. **Case** e **in** Q_{opt}: in this case we include e in the solution, decrement k' by 1, set $G := G - e$, add the color $C(e)$ to C_{used};
   1.2. **Case** e **is not in** Q_{opt}: set $G := G - e$;

Figure 5: Branching on the edges incident on I_{in} whose endpoint in S is labeled IN_{used} or IN.

Proposition 4.19. *The number of paths in the search tree corresponding to the algorithm* **Algo-II** *is at most* $(2e)^k k^{k+1}$*, where e is the base of the natural logarithm.*

Proof. By Proposition 4.18, there are at most k vertices in I_{in}. For each vertex in I_{in}, by Observation 4.17, there are at most $2k$ edges incident on it whose endpoints in S are labeled IN_{used} or IN. Therefore, The number of edges between I_{in} and vertices in S of label IN_{used} or IN is at most $2k^2$.

The branching in the algorithm **Algo-II** can be implemented as follows. Since at most k edges can be in Q_{opt}, we can try every set of at most k edges between I_{in} and the vertices in S of label IN_{used} or IN. The number of such possible sets is at most:

$$\sum_{i=1}^{k} \binom{2k^2}{i} \leq k\binom{2k^2}{k} \leq (2e)^k k^{k+1}.$$

The last inequality is obtained using Stirling's approximation [7]. □

After branching on the edges incident on the vertices in I_{in} and removing them, the vertices in I_{in} and the vertices in S of label OUT can be removed. Every remaining vertex in S is either of label IN_{used} or IN. Since a vertex in S of label IN_{used} is an endpoint of an edge already in Q_{opt}, every edge incident on a vertex in IN_{used} is dominated. Therefore, if for every vertex of label IN in S we determine one of its incident edges to be in Q_{opt}, we obtain an edge dominating set of B. On the other hand, our branching stipulates that from every vertex in S of label IN we must determine at least one edge incident on it to be in Q_{opt}. Therefore, our problem reduces to picking for every vertex of label IN in S exactly one edge incident on it, so that the total number of colors used is minimized. (That is, we do not need to be concerned about which edges an edge incident on a vertex in S dominates, since picking an incident edge from every vertex remaining in S guarantees that we end up with an edge dominating set; the problem thus reduces to picking a set of edges incident on the remaining vertices in S that uses the minimum number of colors.) To do so, we first remove the vertices of label IN_{used} from S, since no edge incident on any of them needs to be considered. At this point S should have at most k' vertices; otherwise, we can reject. Then for every color c in C_{used}, and for every vertex v of label IN in S, if there is an edge of color c incident on v, we include e in the solution, decrement the parameter, and remove the vertex from B. (Note that edges whose color is in C_{used} are "gained for free" because their colors have already been used by edges in the partial solution.)

After this step, every vertex in S is of label IN, and there is no edge incident on any vertex in S whose color appears in C_{used}. To compute a set of edges that uses the minimum number of colors such that for every vertex in S exactly one edge in this set is incident on it, we have the following proposition:

Proposition 4.20. *A set of edges that uses the minimum number of colors and such that, for every vertex in S, exactly one edge incident on it is in this set, can be computed by an algorithm whose corresponding search tree has at most k^{k+1} paths.*

Proof. We try each partition of S into ℓ groups, $\ell \in \{1, \dots, k'\}$, such that all vertices in the same group are incident on edges of the same color in Q_{opt} (as we did in Stage 2 of the MLMM problem). For each such partition, and for each group in this partition, we find a color c such that every vertex in this group is incident on an edge of color c; we add this set of edges of color c to the partial solution. If such a choice is not possible for some group, then we reject the partition. Note that computing such a set of edges can be done in time $O(kn(G))$, because the number of colors is at most $n(G)$ (for every color, and every group, we try whether there is a monochromatic set of edges incident on the group vertices). It is clear that at least one partition will correspond to the same partition of vertices induced by Q_{opt}, and hence, at least one search path will lead to an optimal solution.

Since S has at most k' vertices at this point, the total number of partitions of S is at most $k'^{k'+1} \leq k^{k+1}$. □

Theorem 4.21. Minimum Label Edge Dominating Set *can be solved in time $O(256^k e^{2k} k^{3k+3}(n(G) + e(G)))$, and hence is FPT when parameterized by the size of the edge dominating set.*

Proof. We apply **Algo-I**, followed by **Algo-II**, followed by the algorithm described in Proposition 4.20. At the end, we end up with an edge dominating set for G of size at most k. We output the edge dominating set of G of size at most k that uses the minimum number of colors, over all solutions generated from all branches.

By Proposition 4.16, Proposition 4.19, and Proposition 4.20, the total number of partial solutions enumerated by the algorithm is $O((128ek)^k \cdot (2e)^k k^{k+1} \cdot k^{k+1}) = O(256^k e^{2k} k^{3k+2})$. For each such partial solution, we need to process the graph G during the branching, which takes time $O(kn(G) + e(G))$. Note that the $kn(G)$ factor is the time needed to compute the sets described in Proposition 4.20. Therefore, the running time of the algorithm is $O(256^k e^{2k} k^{3k+3}(n(G) + e(G)))$.

It follows that the Minimum Label Edge Dominating Set problem is FPT when parameterized by the size of the edge dominating set. □

5. Concluding Remarks

In this paper, we considered some minimum label graph problems. We showed that, when parameterized by the number of used labels, most of these problems are intractable, even on graphs of bounded pathwidth. On the other hand, we showed that most of these problems become parameterized tractable when parameterized by the solution size. For the tractability results developed in this paper, the parameterized algorithms we presented are not very practical, and improving these algorithms is definitely possible, and remains an open question. In particular, the following questions stand out, following established lines of investigation in parameterized algorithmics [12]:

1. Are there FPT algorithms for the FPT problems we have identified here, having runtime of the form $2^{O(k)} n^{O(1)}$?

2. Do these FPT problems admit polynomial-time kernelization to problem kernels of size bounded by a polynomial in the parameter k (see [1])?

We note that, recently, there has been a lot of interest in studying structured graph problems, such as problems on colored graphs, due to their applications in various fields such as networking and computational biology. (The CONVEX RECOLORING problem [17] is such an example in computational biology.) We also note that certain genetic phase solution recombination problems, in the setting of meta-heuristics for hard problems, can be formulated as maximum-label graph problems [18]. While these problems are practically very important, they are often computationally hard due to the structural requirement on the solution sought. Therefore, it is both natural and interesting to study whether these problems remain intractable with respect to different parameters, such as the number of colors, the pathwidth/treewidth of the graph, the solution size, or even with respect to more restrictive parameters, such as the vertex cover or the max leaf number. This paper follows this line of research [10].

Finally, it is interesting to study the parameterized complexity of other minimum label graph problems that have practical applications. A good candidate would be the Minimum Label Feedback Arc Set problem on directed graphs.

Acknowledgments: We would like to thank the anonymous referee for the valuable comments and suggestions that helped improve the quality and the presentation of the paper.

References

[1] H. Bodlaender, R. Downey, M. Fellows, and D. Hermelin. On problems without polynomial kernels. *J. Comput. Syst. Sci.*, 75(8):423–434, 2009.

[2] H. Broersma and X. Li. Spanning trees with many or few colors in edge-colored graphs. *Discussiones Mathematicae Graph Theory*, 17(2):259–269, 1997.

[3] H. Broersma, X. Li, G. Woeginger, and S. Zhang. Paths and cycles in colored graphs. *Australasian Journal on Combinatorics*, 31:299–311, 2005.

[4] T. Brüggemann, J. Monnot, and G. Woeginger. Local search for the minimum label spanning tree problem with bounded color classes. *Operations Research Letters*, 31(3):195–201, 2003.

[5] R. Carr, S. Doddi, G. Konjevod, and M. Marathe. On the red-blue set cover problem. In *Proc. 11th ACM-SIAM SODA*, pages 345–353, 2000.

[6] R. Chang and S. Leu. The minimum labeling spanning trees. *IPL*, 63(5):277–282, 1997.

[7] T. Cormen, C. Leiserson, R. Rivest, and C. Stein. *Introduction to Algorithms, 2nd edition.* McGraw-Hill Book Company, Boston, MA, 2001.

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

[9] M. Fellows, D. Hemmelin, and R. Rosamond. On the fixed-parameter intractability and tractability of multiple-interval graph properies. *Theoretical Computer Science*, 410:53–61, 2009.

[10] M. Fellows, D. Lokshtanov, N. Misra, M. Mnich, F. Rosamond, and S. Saurabh. The complexity ecology of parameters: An illustration using bounded max leaf number. *Theory of Computing Systems*, 45:822–848, 2009.

[11] R. Hassin, J. Monnot, and D. Segev. Approximation algorithms and hardness results for labeled connectivity problems. *Journal of Combinatorial Optimization*, 14(4):437–453, 2007.

[12] F. Hüffner, R. Niedermeier, and S. Wernicke. Techniques for practical fixed-parameter algorithms. *Comput. J.*, 51(1):7–25, 2008.

[13] S. Jha, O. Sheyner, and J. Wing. Two formal analyses of attack graphs. In *Proc. 15th IEEE Computer Security Foundations Workshop*, pages 49–63, 2002.

[14] Ton Kloks. *Treewidth, Computations and Approximations*, volume 842 of *Lecture Notes in Computer Science.* Springer, 1994.

[15] S. Krumke and H. Wirth. On the minimum label spanning tree problem. *IPL*, 66(2):81–85, 1998.

[16] J. Monnot. The labeled perfect matching in bipartite graphs. *IPL*, 96(3):81–88, 2005.

[17] S. Moran and S. Snir. Convex recolorings of strings and trees: Definitions, hardness results and algorithms. *Journal of Computer and System Sciences*, 74(5):850–869, 2008.

[18] P. Moscato. Personal communication.

[19] T. Uno. Algorithms for enumerating all perfect, maximum and maximal matchings in bipartite graphs. In *Proc. 8th ISAAC*, volume 1350 of *LNCS*, pages 92–101. Springer, 1997.

[20] S. Voss, R. Cerulli, A. Fink, and M. Gentili. Applications of the pilot method to hard modifications of the minimum spanning tree problem. In *Proc. 18th MINI EURO Conference on VNS*, 2005.

[21] Y. Wan, G. Chen, and Y. Xu. A note on the minimum label spanning tree. *IPL*, 84(2):99–101, 2002.

[22] D. West. *Introduction to graph theory*. Prentice Hall Inc., Upper Saddle River, NJ, 1996.

[23] H. Wirth. *Multicriteria Approximation of Network Design and Network Upgrade Problems*. PhD thesis, Department of Computer Science, Universität Würzburg, Germany, 2005.

[24] Y. Xiong. *The Minimum Labeling Spanning Tree Problem and Some Variants*. PhD thesis, Graduate School of the University of Maryland, USA, 2005.

[25] Y. Xiong, B. Golden, and E. Wasil. A one-parameter genetic algorithm for the minimum labeling spanning tree problem. *IEEE Transactions on Evolutionary Computation*, 9(1):55–60, 2005.

[26] Y. Xiong, B. Golden, and E. Wasil. Worst case behavior of the MVCA heuristic for the minimum labeling spanning tree problem. *Operations Research Letters*, 33(1):77–80, 2005.

[27] P. Zhang, L. Tang, W. Zhao, J. Cai, and A. Li. Approximation and hardness results for label cut and related problems, 2007. Manuscript.