ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Algorithms
and Didactics

Maximum Matching with Disjunctive Constraints

Leyla Latifova

Bachelor's Thesis

2025

Supervisors: Prof. Dr. D. Komm
Dr. H.-J. Böckenhauer
P. Whittington

Abstract

In this thesis, we study the problem first introduced by Darmann et al. [1], which deals with the Maximum Matching problem under additional constraints between any two arbitrary edges in the graph. The constraints can either forbid the two edges to be together in a valid solution (conflict constraints), or require for at least one of the two to be included (forcing/covering constraints). The constraints are typically encoded as part of the input in a constraint graph.

For both variants we complement the previously known NP-hardness results and show that the problems on paths with constraint graphs with maximum degree 1 are NP-hard.

For the conflict case, we show that the problem is W[1]-hard when parameterized by the size of the maximum matching, and for the covering variant, we present an $\mathcal{O}^*(2^k)$ algorithm, where k is the output solution size. We additionally introduce an extended constraint graph, which allows for FPT algorithms parameterized by its treewidth, one for each variant.

Additionally, we present a 2-approximation algorithm for general unweighted covering variant, and generalize the previously known result to get an approximation algorithm for the conflict matching in terms of the maximum number of constraints an edge participates in.

Acknowledgements

I would like to thank my supervisors, family, friends, Shustrik, Bublik, and Masyanya.

Contents

Chapter 1

Introduction

In recent years, the study of optimization problems under additional constraints has gained attention in complexity theory. This typically involves computing a solution to a problem which needs to satisfy some given conditions. In this work we focus on the Maximum Matching problem with additional constraints between any two edges of the input graph. We look at cases where we can forbid any two edges to be simultaneously in a valid solution (conflict constraints), or enforce at least one of the two to be included (forcing/covering constraints). The conventional Maximum Matching problem can be solved in polynomial time, however, becomes NP-hard when extended with such constraints. They are also called *disjunctive*, as they can be represented through disjunctions.

Formally, an instance for the Conflict-free Maximum Matching problem, or CF-MM, is a pair of two graphs (G, H), where a valid solution is a matching in G and an independent set in H. For forcing constraints, an instance for the Covering Maximum Matching, or Cov-MM, is also a pair of two graphs (G, H), where a valid solution is a matching in G and a vertex cover in H. For both of these variants, the vertex set of H is the edge set of G, and the goal is to maximize the size of the matching which satisfies the constraints. These are encoded in H, which is generally referred to as the *constraint graph*. In case of CF-MM, it is called the *conflict graph*, and as the *forcing graph* for the covering variant. We also consider the weighted variants of the problems, where weights are assigned to the edges and the goal becomes to find a matching which satisfies the constraints with the largest weight. The instances for the weighted case are denoted as (G, H, c) with c being the weight function on E_G.

Important to note here is that CF-MM and Cov-MM is not a notation that was used when the problem was introduced. CF-MM however is used in the latest literature.

Here we focus on the classical computational, parameterized, and approximation complexity of both variants. Darmann et al. [1] have shown that CF-MM and Weighted Cov-MM are NP-hard on a disjoint union of rectangles with a 1-regular constraint graph. We extend this result further and show that both of the variants are NP-hard on a path with a 1-regular conflict/forcing graphs in the unweighted case. For the Cov-MM case, even restricting the forcing graphs to those of regular degree still leaves the subproblem NP-hard. We also show that the Cov-MM problem is in P in case the maximum matching of the forcing graph is at least as large as the

maximum matching size of the original graph for the matching.

When it comes to the parameterized complexity of the conflict variant, Agrawal et al. [2] have studied the problem extensively, and have shown that the general unweighted version is W[1]-hard with respect to solution size. We show that the problem is also W[1]-hard when parameterized by the maximum matching of the graph for a matching. The authors also present FPT algorithms for restricted input graphs, such as when the conflict graph is chordal or d-degenerate. The forcing variant, however, has not been touched upon before. We show that Cov-MM admits a simple algorithm running in $\mathcal{O}^*(2^k)$, where k is the output solution size, and prove that an FPT algorithm running in time $2^{o(k)} \cdot n^c$ does not exist, unless ETH fails. We additionally introduce the extended constraint graph, which allows richer parameterization for both CF-MM and Cov-MM.

Compared with classical computational and parameterized complexity, the problem has not been studied extensively from the viewpoint of approximation. Darmann et al. [1] have introduced a 2-approximation for the weighted CF-MM when the conflict graph has maximum degree of 1. We generalize this result and introduce a $(\Delta_H + 1)$-approximation algorithm for the weighted case, where Δ_H is the maximum degree of the conflict graph. For the forcing variant we show that the problem of deciding the feasibility of the covering matching is in P, and utilize it to design a 2-approximation algorithm for the general unweigthed Cov-MM.

Chapter 2

Preliminaries

2.1 General definitions and notations

We denote $\mathbb{N}$ as the set of natural numbers without 0, and $\mathbb{N}_0$ as the one with zero. The set of integers from 1 to n is denotes as $[n]$, where $n \in \mathbb{N}$.

Here we consider only simple undirected graphs. A *graph* G is a tuple $G := (V, E)$, where V is the vertex and E is the edge set. To refer to the vertex/edge sets of a graph G, we write V_G and E_G. An (undirected) *edge* between two distinct vertices u and v is denoted as $\{u, v\}$. u and v in this case are called the *endpoints* of $\{u, v\}$. Two edges e_1, e_2 are *adjacent* if $e_1 \neq e_2$ and $e_1 \cap e_2 \neq \emptyset$. Two vertices $u, w \in V$ are *adjacent* if there exists an edge $\{u, w\}$ in E. A vertex $u \in V$ is *incident* to an edge e if $u \in e$.

The *open neighbourhood* of a vertex $u \in V$ in G is defined as the set of all adjacent vertices to u, and is denoted by $N_G(u)$. Formally, $N_G(u) := \{w \mid \{u, v\} \in E\}$. $N_G[u]$ denotes the *closed neighbourhood* of a vertex $u \in V$ in G and is defined as its open neighbourhood with u itself, namely $N_G[u] := N_G(u) \cup \{u\}$. The *degree* of a vertex u in G is defined as the size of its open neighbourhood in G, denoted as $\deg_G(u)$. Whenever the graph is clear from the context, the degree of v is denoted as $\deg(v)$. An edge $\{u, w\} \in E$ is called a *leaf edge*, if either $\deg(u) = 1$ or $\deg(w) = 1$. An edge $\{u, w\}$ is *isolated*, or *independent*, if $\deg(u) = \deg(w) = 1$. For $k \in \mathbb{N}_0$, a graph G is called *k-regular*, if for every vertex $v \in V_G$, $\deg(v) = k$.

A *subgraph* $G' := (V', E')$ of G is a graph such that $V' \subseteq V$, $E' \subseteq E$ and $\bigcup_{e' \in E'} e' \subseteq V'$. A *vertex-induced subgraph* $G' := (V', E')$ is defined by its vertex set $V' \subseteq V$, such that $E' := \{\{u, w\} \mid u \in V', v \in V', \{u, w\} \in E\}$, and is denoted as $G[V']$. An *edge-induced subgraph* $G' := (V', E')$ is defined by its edge set $E' \subseteq E$, such that $V' := \bigcup_{e' \in E} e$, and is denoted as $G[E']$.

A *path of length* $n \in \mathbb{N}_0$, denoted as P, is an ordered sequence of $n + 1$ distinct vertices, denoted as $(u_1, \ldots, u_n, u_{n+1})$, such that $\{u_i, u_{i+1}\} \in E$ for all $1 \leq i \leq n$. We say that P is a path *from* u_1 *to* u_{n+1}, or *between* u_1 *and* u_{n+1}, and that u_1 and u_{n+1} are *connected*. A *shortest path* between u an v in G is defined as a path between u and v of the smallest length. A graph G is *connected* if for every $u, w \in V$ there exists a path from u to w. A *cycle* of length $n \in \mathbb{N}$ with $n \geq 3$ is an ordered sequence of $n + 1$ vertices $(u_1, \ldots, u_n, u_{n+1})$, such that $u_1 = u_{n+1}$, $|\{u_1, \ldots, u_{n+1}\}| = n$, and $\{u_i, u_{i+1}\} \in E$ for all $1 \leq i \leq n$.

A graph G is called *acyclic* or a *forest* if there does not exist a set $V' := \{u_1, \dots u_n\} \subseteq V$ for some $n \in \mathbb{N}$, such that $(u_1, \dots, u_n, u_1)$ is a cycle in G.

A graph is a *tree* if it is acyclic and connected. A tree T with a designated *root vertex* r is called an *r-rooted tree* or just a *rooted tree*. The *depth* of a vertex u in an *r-rooted tree* T is defined as the length of the path from u to r. The *height* of a rooted tree is defined as the maximum depth of all the vertices in the tree. A *leaf vertex* of a tree is a vertex with degree 1.

The *complement graph* of G is denoted as $G^c := (V^c, E^c)$, where $V^c = V$ and $E^c := \{\{u, w\} \mid u, w \in V,\ \{u, w\} \notin E\}$. A set $V' \subseteq V$ is an *independent set* in G if the edge set of the vertex-induced subgraph, namely $G[V']$, is empty. A set $V' \subseteq V$ is a *clique* in G if V' is an independent set in G^c. A set $V' \subseteq V$ is a *vertex cover* of G if, for every edge $e \in E$, $e \cap V' \neq \emptyset$.

Generally, for a problem U on a graph G, such that its solutions are sets, a valid solution C is *maximal* if there is no valid solution C', such that $C \subset C'$. A solution C is called *maximum* if, for all valid solutions C', $|C'| \leq |C|$. In turn, a solution C is called *minimal* if there does not exist a valid solution C' such that $C' \subset C$, and *minimum* if, for all valid solutions C', $|C| \leq |C'|$.

A set $M \subseteq E$ is a *matching* of G if all edges in M are not pairwise adjacent in G. We denote $M(G)$ as the size of the maximum matching of G.

$\mathcal{C}_n$ denotes a class of cycles of length n with $n \geq 3$, and $\mathcal{C} := \bigcup_3^\infty \mathcal{C}_n$. $\mathcal{P}_n$ denotes a class of paths of length n for $n \geq 0$, and $\mathcal{P} := \bigcup_0^\infty \mathcal{P}_n$. The class of graphs which are disjoint unions of paths is denoted as $\mathcal{P}^{\mathcal{U}}$. For $k \geq 0$, the class of graphs with maximum degree k is denoted as Δ_k and the class of k-regular graphs is denoted as $\mathcal{R}_k$.

We refer to the general unweighted Conflict-free Maximum Matching as CF-MM and general unweighted Covering Maximum Matching as Cov-MM. Both problems with instances (G, H) restricted only to those with G belonging to a graph class $\mathcal{D}_1$ and H to $\mathcal{D}_2$ are referred to as CF-MM-$(\mathcal{D}_1, \mathcal{D}_2)$ and Cov-MM-$(\mathcal{D}_1, \mathcal{D}_2)$. In case G or H are not restricted to a graph class, $\mathcal{G}$ is used instead of $\mathcal{D}_1$, and $\mathcal{H}$ instead of $\mathcal{D}_2$ respectively.

We use n with a different meaning depending on the context. Whenever we are dealing with a non-graph problem, n refers to the instance length. When considering a graph problem other than CF-MM or Cov-MM, we use n for the number of vertices in the input graph. In the case of CF-MM and Cov-MM, n refers to the number of vertices in G for an instance (G, H).

$\mathcal{O}^*(f(k))$ is short for $\mathcal{O}(f(k) \cdot p(n))$, where f and p are functions and p is additionally a polynomial.

2.2 Parameterized Complexity definitions

Here we base the majority of the definitions on the books “Parameterized algorithms” by Cygan et al. [3], "Kernelization" by Fomin et al. [4], and “Algorithmics for Hard Problems” by Hromkovič [5].

Definition 2.1 (Fixed parameter tractability). Let $L \subseteq \Sigma^*$ be the language of a decision problem U and $par\colon L \mapsto \mathbb{N}$ a computable function (called *parameteriza-*

tion), such that, for infinitely many $k \in \mathbb{N}$, the set

$$\text{set}_U(k) := \{I \in L \mid par(I) = k\}$$

is infinite. A *parameterized problem*, characterized by a tuple (L, par), is a subset $(L, par) \subseteq \Sigma^* \times \mathbb{N}$, such that $(I, k) \in (L, par)$ iff $I \in L$ and $par(I) = k$.

An algorithm $\mathcal{A}$, which for an instance $I \in \Sigma^*$ decides whether $I \in L$ in time $f(par(x)) \cdot (|I|)^{\mathcal{O}(1)}$, where f is a computable function, is called a *par-parameterized algorithm for U*. In case such an algorithm exists, we call U *fixed-parameter tractable* (or *fpt*) *according to*, or *with respect to par*. The class of all fixed-parameter tractable problems is denoted as FPT.

Definition 2.2 (Parameterized reduction). Let U_1 and U_2 be two parameterized problems. A *parameterized* or an *fpt reduction* from U_1 to U_2 is an algorithm $\mathcal{A}$ that, given an instance (I_1, k_1) for U_1, computes an instance (I_2, k_2) for U_2, such that the following hold

1. $(I_1, k_1) \in U_1$ if and only if $(I_2, k_2) \in U_2$
2. $k_2 \leq g(k_1)$ for some computable function $g \colon \mathbb{N} \mapsto \mathbb{N}$
3. $\mathcal{A}$ runs in time at most $f(k_1) \cdot (|I_1|)^{O(1)}$, for some computable function f

In case such a parameterized reduction exists, we say that U_1 is *fpt-reducible* to U_2, denoted $U_1 \leq_{fpt} U_2$.

Parameterized reductions are very useful to prove that some problems are in FPT without designing an algorithm explicitly. This is due to the following result:

Lemma 2.1 (Lemma 2.2, [6]). FPT is closed under fpt reductions.

There is an alternative way to look at problems in FPT as well.

Definition 2.3 (Kernelization). An algorithm $\mathcal{A}$ is a *kernelization algorithm*, or a *kernel*, for a parameterized problem U, if given an instance (I_1, k_1) of U, returns an instance (I_2, k_2) of U in time $|I_1|^c$ for some $c \in \mathbb{N}$, such that $(I_1, k_1) \in U$ if and only if $(I_2, k_2) \in U$ and $|I_2| + k_2 \leq f(k_1)$ for some computable function f. The *size of the kernel* is defined by the function f; if f is polynomial, then we say that the kernel is of polynomial size.

An existence of a kernelization algorithm for a parameterized problem U implies that the problem is in FPT, considering that one can run the kernelization algorithm and perform exhaustive search on the returned instance. Along with Theorem 1.4 ([4]), we have the following result.

Corollary 2.1. A parameterized problem U is in FPT if and only if it admits a kernelization algorithm.

We also define a similar notion of hardness and completeness for parameterized problems as in classical complexity theory.

Definition 2.4 (Parameterized complexity hardness and completeness). For a class of parameterized problems C, a parameterized problem U is *C-hard* if all problems in C are fpt-reducible to it. U is also *C-complete* if it is in C and is C-hard.

Next we introduce a simplified and equivalent to the initial definition of the class W[1]. The original definition is more involved and for the sake of simplicity not introduced here. It is believed that FPT $\subset$ W[1], namely that W[1]-hard problems do not have a parameterized algorithm.

Definition 2.5 (W[1]). A parameterized problem U is in W[1] if is fpt-reducible to the Independent Set Problem parameterized according to the solution size.

Next we define the class para-NP, which plays an analogous role to FPT as NP does in relation to P in the classical complexity theory. In fact, para-NP = FPT if an only if P = NP. We give only the basic definition and one consequent result.

Definition 2.6 (para-NP). A parameterized problem $(U, par) \subseteq \Sigma^* \times \mathbb{N}$ is in *para-NP*, if there is a computable function $f\colon \mathbb{N} \mapsto \mathbb{N}$ and a non-deterministic algorithm that, given an instance $I \in \Sigma^*$, decides whether $(I, par(k)) \in (U, par)$ in time at most $f(par(I)) \cdot (|I|)^{\mathcal{O}(1)}$.

A problem in NP is automatically in para-NP with respect to any parameterization. Consequently, we have the following implication.

Corollary 2.2. Let (U, par) be a parameterized problem in para-NP. If there exists a constant $c \in \mathbb{N}$, such that

$$(U, par)_c := \{I \in U \mid par(I) = c\}$$

is NP-hard, then (U, par) is para-NP-hard.

The class para-NP is a superset of the class W[1] (Proposition 3.2, [6]), and we thus have the following result.

Corollary 2.3. Every para-NP-hard problem is W[1]-hard.

A familiar example would be the k-Coloring Problem parameterized by k, which is already NP-hard for $k = 3$.

We also outline the definition of a tree decomposition; a concept proven to be quite useful for parameterized problems.

Definition 2.7 (Treewidth). A *tree decomposition* of a graph G is a pair (X, T), where the elements of X are subsets of V_G (called *bags*), and T is a tree whose nodes are the elements of X, for which the following holds:

1. $\bigcup_{X_v \in X} V_G$
2. For every $\{u, v\} \in E_G$ there exists a bag X_w, such that $\{u, v\} \subseteq X_w$
3. For every $u \in V_G$, $T[T_u]$ is connected, where $T_u := \{X_w \in V_T \mid u \in X_w\}$

The *width* of $\mathcal{T}$ is $\max_{X_v \in X}\{|X_v| - 1\}$. The *treewidth* of a graph G is the minimum width of all the tree decompositions of G, and is denoted by $tw(G)$.

We also outline the *Exponential time hypothesis*, which we will not use explicitly, but rather show that certain results hold under its assumption.

Conjecture 2.1 (Exponential time hypothesis/ETH). There is no $2^{o(n)}$ algorithm for 3SAT.

Chapter 3

Classical Computational Complexity

First, we analyze the two problems using traditional computational complexity. We complement the previous results obtained by Darmann et al. [1] and show that even such a simple graph as a path with each edge participating in at most one constraint still makes the two problems NP-hard.

We additionally examine the feasibility of Cov-MM, show that it is in P, and apply that result to obtain a 2-approximation algorithm. Furthermore, we experiment with the number of constraints and improve the lower bound on the number of constraints with which the problems are NP-hard.

Before we show the NP-hardness result on paths for CF-MM, we first show that it is NP-hard on a disjoint union of paths, denoted as $\mathcal{P}^{\mathcal{U}}$.

3.1 CF-MM

Lemma 3.1. CF-MM-$(\mathcal{P}^{\mathcal{U}}, \Delta_1)$ is NP-hard.

Proof. We proceed by reducing from the Vertex Cover Problem.

Let $G := (V, E)$ be the input graph for the Vertex Cover with n vertices. We construct the input instance (G', H') for the CF-MM-$(\mathcal{P}^{\mathcal{U}}, \Delta_1)$ in the following way. For every vertex $v \in V$, create a path of $2n - 1$ edges e_i for $1 \leq i \leq 2n - 1$. We divide the edges into positive and negative, where positive edge are those with an even index, and negative with an odd index. The idea is that positive edges would represent the vertex being included in the vertex cover, while negative edges are for vertices that are excluded. We refer to the j-th positive and negative edge of vertex v as v_j and $\bar{v}_j$ respectively. With an arbitrary ordering of neighbours for each vertex, for an edge $\{u, v\} \in E$, add a constraint $\{\bar{u}_i, \bar{v}_j\}$ if u is the j-th neighbour of v and v is the i-th neighbour of u. An example of the reduction is illustrated in Figure 3.1. Now we show the following:

For $k \leq n$:
There exists a vertex cover of G with size at most $k \iff$ There exists a conflict-free matching M in (G', H') of size at least $n^2 - k$.

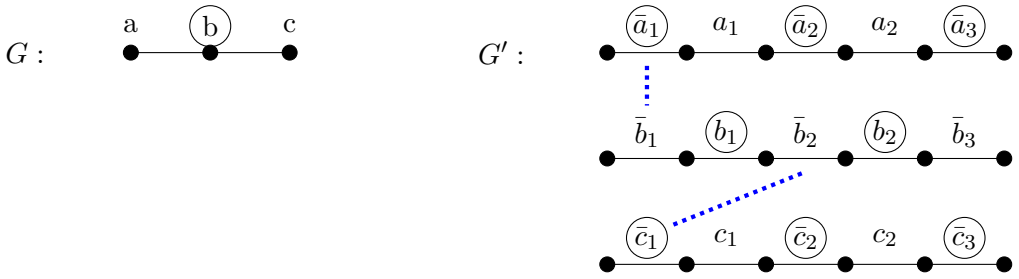


Figure 3.1. An example of the reduction in Lemma 3.1. The black lines indicate the edges and the blue dashed lines represent the constraints. The vertices of the minimum vertex cover of G of size 1, which is $\{b\}$, as well as corresponding conflict-free matching edges (total of $n^2 - 1 = 8$) in of G' are in circles.

($\Rightarrow$) Let C be a vertex cover of G with size at most k. For $v \in C$, add the positive edges v_q (for $1 \leq q \leq n-1$) to the matching M. For $u \notin C$, add the negative edges v_q (for $1 \leq q \leq n$) to the matching. This would result in a matching of size at least $k \cdot (n-1) + (n-k) \cdot n = n^2 - k$. All of the edges in M are non-adjacent, and thus M is a valid matching. For a constraint $\{\bar{u}_i, \bar{v}_j\}$ in H, by construction there exists an edge $\{u, v\}$ in G. Since C is a vertex cover, at least one of u and v is in C. By construction of the matching, for a vertex in C, only its positive edges are included. This ensures that either $\bar{u}_i$ or $\bar{v}_j$ are not in the matching, therefore not violating the constraint. Consequently, M is a conflict-free matching.

($\Leftarrow$) Assume M is a matching of size at least $n^2 - k$. Then there must be at least $n - k$ vertices which have all negative edges in the matching. Otherwise, the matching has size at most $(n-k-1) \cdot n + (k+1) \cdot (n-1) = n^2 - k - 1 < n^2 - k$. For all the vertices which do not have all negative edges matched (at most k of those), add them to the set C. Then for a vertex $v \notin C$, by definition of C all of its negative edges must be in M. By definition of constraints, for an edge $\{v, w\}$ in G, one negative edge of v must be in a constraint with one negative edge of w. As all of the negative edges of v are in M, that means the corresponding negative edge of w cannot be in M, and thus w must be in C. As a result of this observation, for all non-isolated vertices not in C, its adjacent vertices are in C, which implies that C is a vertex cover in G. □

One can further use the reduction to show that the problem is NP-hard on instances (G, H) where G is a single path and every edge participates in at most one conflict. A similar technique is outlined in the proof of Lemma 3.5. However, we take a different route by first showing that the problem is NP-hard on a cycle with a 1-regular conflict graph by reducing from Maximum Independent Set on 3-regular Hamiltonian graphs, which is NP-hard. Later on, we break down a cycle into a path. But first we show that the problem is NP-hard on cycles.

Lemma 3.2. CF-MM-$(\mathcal{C}, \mathcal{R}_1)$ is NP-hard.

The proof proceeds by breaking down a 3-regular Hamiltonian graph into a 2-regular and a 1-regular component. This is done using the Hamiltonian cycle, which "becomes" a cycle, and the rest of the graph, which is 1-regular, becomes the conflict graph. An example for a reduction is given in Figure 3.2. For the sake of simplicity, the

example is not a result of the reduction given by Fleischner et al. [7].

Proof. We use a reduction from the Maximum Independent Set on graphs in PH_3, a class of planar 3-regular Hamiltonian graphs, which was shown to be NP-hard by Fleischner et al. [7]. Since the reduction in the paper involves constructing a graph with a clearly outlined Hamiltonian cycle, let G and $Q := (1, 2, \ldots, n, 1)$ be the respective input graph and cycle (in the paper G' is used as the notation for the Hamiltonian graph which is a result of their reduction).

Construct a graph G' and a conflict graph H' in the following way.

- $V_{G'} := \{v_{i,j} \mid 1 \leq i \leq n,\ j = (i \bmod n) + 1\}$. The vertex is defined for each pair of edges which appear consecutively in cycle Q.
- $E_{G'} := \{e_i := \{v_{j,i}, v_{i,k}\} \mid 1 < i < n,\ j = (i-1),\ k = i+1\}$ $\bigcup \left\{e_1 := \{v_{n,1}, v_{1,2}\},\ e_n := \{v_{(n-1),n}, v_{n,1}\}\right\}$. Essentially, e_i is an edge representation of the vertex i.
- $V_{H'} := E_{G'}$
- $E_{H'} := \{\{e_i, e_j\} \mid (i-j) \bmod n > 1,\ \{i, j\} \in E_G\}$. The constraints are only between adjacent edges which do not appear consecutively in the cycle Q. There is exactly one for each edge, since two of the neighbours of a vertex are adjacent in the cycle and G is 3-regular, which leaves exactly one out.

Then for $0 \leq k \leq |E_G|$:

There exists an independent set of size at least k in $G \iff$ There exists a conflict-free matching in (G', H') of size at least k

($\Rightarrow$): Let I be an independent set in G' of size at least k. For every $v \in I$, add e_v into a set M. The cardinality of M is thus at least k. M is a matching because for any two vertices $u, v \in I$, the edges e_u and e_v are not adjacent, as edges only of adjacent vertices in G are also adjacent in G'. M is also conflict-free, because conflicts between two edges are present only if the two corresponding vertices are adjacent in G. Thus, no two edges in M conflict as all vertices in I are pairwise non-adjacent.

($\Leftarrow$): Let M be a conflict-free matching in (G', H') of size at least k. For every $e_u \in M$, add u into a set I. The cardinality of I is thus at least k. I is an independet set in G, because for an edge $\{u, v \in E_G\}$, either e_u is adjacent to e_v, in which case the two cannot be simultaneously in a valid matching, or e_u is in conflict with e_v, and the two cannot be simultaneously in a conflict-free matching. Since M is a conflict-free matching, neither of the two cases occur, and I is indeed an independent set. □

We use the previous result to show a stronger one on a path. The idea is to simulate a cycle using a path by cutting it at an edge and appropriately introducing additional constraints. To be precise, we pick two adjacent edges in a path, split the cycle at that point, add one edge at each end of the resulting path, and put the new edges together in a conflict. An example of the reduction is shown in Figure 3.3.

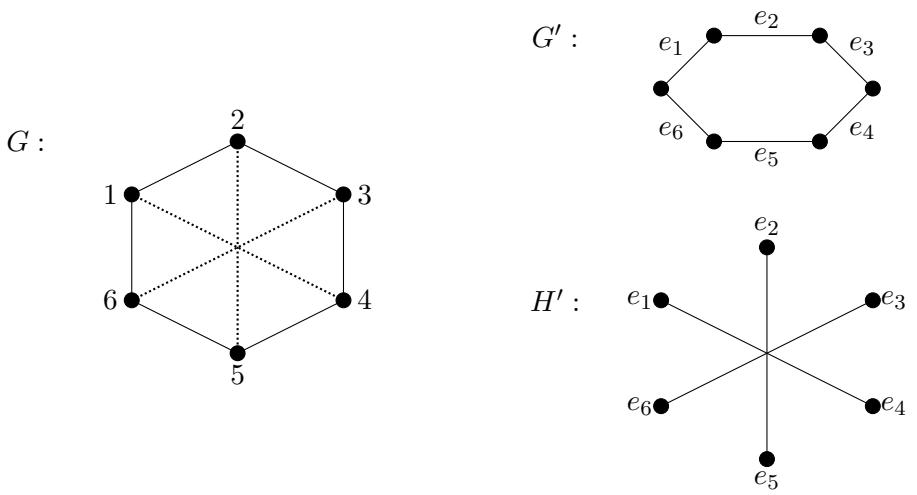


Figure 3.2. An example of the reduction in Lemma 3.2. The dotted lines are also edges of G; they are isomorphic to H'. The maximum independent set in G is of size 3, e.g., the set $\{a, c, e\}$. It is also the case that it is one of the maximum conflict-free matchings of (G', H').

Lemma 3.3. CF-MM-$(\mathcal{P}, \Delta_1)$ is NP-hard.

Proof. We proceed with a reduction from CF-MM-$(\mathcal{C}, \Delta_1)$. As a clarification, n here denotes the number of vertices in the input graph, but we also use it to name a vertex. Let (G, H) be the input graphs to the CF-MM-$(\mathcal{C}, \Delta_1)$ problem with n vertices, where

- $V_G := \{v_1, \ldots, v_n\}$
- $E_G := \{e_i := \{v_i, v_{(i \bmod n)+1}\} \mid 1 \leq i \leq n\}$

Construct graphs G' and H' (conflict graph), where

- $V_{G'} := \{v_0, v_1, \ldots, v_{n+2}\}$
- $E_{G'} := \{e_i := \{v_i, v_{i+1}\} \mid 0 \leq i \leq n+1\}$. We include the old edges, as well as two at each end of a path.
- $V_{H'} := E_{G'}$
- $E_{H'} := E_H \bigcup \{\{e_0, e_{n+1}\}\}$. Here we add a conflict between the two of the new edges.

Then for $0 \leq k \leq |E_G|$:

There exists a conflict-free matching of (G, H) of size at least $k \iff$ There exists a conflict-free matching in (G', H') of size at least $k + 1$

($\Rightarrow$): Let M be a conflict-free matching of (G, H) of size at least k. Construct a conflict-free matching M' of (G', H') by case distinction:

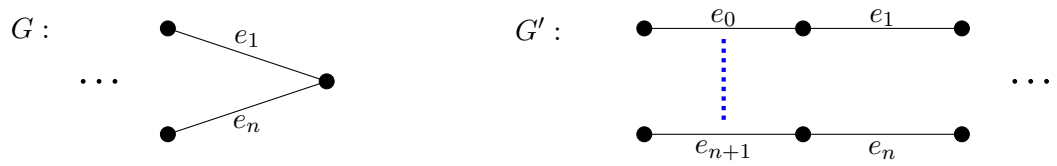


Figure 3.3. An example of the reduction in Lemma 3.3. The blue dashed lines here indicate the new constraints between the edges. The edges e_1 and e_n can still be in a conflict, however, for simplicity not shown here. One can see that from the four edges only 2 can be in the conflict-free matching of the reduced path.

Case 1. $e_1, e_n \notin M$. Then let $M' := M \cup \{e_0\}$.

Case 2. $e_1 \in M$. Then $e_n \notin M$, because M is a valid matching in G. Then let $M' := M \cup \{e_{n+1}\}$.

Case 3. $e_n \in M$. Then $e_1 \notin M$, because M is a valid matching in G. Then let $M' := M \cup \{e_0\}$.

In all 3 cases the size of M' is at least $k+1$, the edges in $M' - M$ are not adjacent to any edges in M, and none of the additional constraints are violated.
($\Leftarrow$) : Let M' be a conflict-free matching of (G', H') of size at least $k+1$. Construct a conflict-free matching M of (G, H) by case distinction:

Case 1. $e_1 \in M'$ and $e_n \notin M'$. Then let $M := M' \cap E_G$. Due to the constraint $\{e_0, e_{n+1}\}$, at most one of the new edges can be in the matching, and thus the size of M is at least k.

Case 2. $e_n \in M'$ and $e_1 \notin M'$. Analogous to the previous case.

Case 3. $e_1, e_n \in M'$. The let $M := M' - \{e_1\}$. As we remove e_1, it is ensured that e_1 and e_n are not simultaneously in M. As the new edges are adjacent to either e_1 or e_n, none of them can be in M', as M' is a matching. Therefore, M only includes edges of G, namely $M \cap E_G = M$, which means it is a valid conflict-free matching in G, as it is a subset of M'. Its size is also at least k, because at most one element is removed. □

3.2 Covering-MM

Now we outline almost identical results to what has been shown previously, but do so for the covering variant. As a reminder, the difference between Cov-MM and CF-MM is that in CF-MM constraints forbid two edges in a constraint to be simultaneously in a valid matching, while the forcing constraints enforce at least one of the two to be included. We start just like previously with showing the problem is NP-hard on a disjoint union of paths with each edge participating in at most one constraint.

Lemma 3.4. Cov-MM-$(\mathcal{P}^{\mathcal{U}}, \Delta_1)$ is NP-hard.

The proof is analogous to the one of Lemma 3.1, with the constraints defined between the positive rather than the negative edges. We outline (almost identical) details here again.

Proof. Let G be the input graph for the Vertex Cover with n vertices. We construct the input instance (G', H') for the Cov-MM-$(\mathcal{P}^{\mathcal{U}}, \Delta_1)$ in the following way. For every vertex $v \in V_G$, create a path of $2n-1$ edges e_i for $1 \leq i \leq 2n-1$. We divide the edges into positive and negative, where positive edges are those with an even index, and negative with an odd index. The idea is that positive edges would represent the vertex being included in the vertex cover, while negative edges are for vertices that are excluded. We refer to the j-th positive and negative edge of vertex v as v_j and $\bar{v}_j$ respectively. With an arbitrary ordering of neighbours for each vertex, for an edge $\{u, v\} \in E_G$, add a constraint $\{u_i, v_j\}$ if u is the j-th neighbour of v and v is the i-th neighbour of u. An example of the reduction is illustrated in Figure 3.4. Now we show the following:

For $k \leq n$:
There exists a vertex cover of G with size at most $k \iff$ There exists a covering matching M in (G', H') of size at least $n^2 - k$.

($\Rightarrow$) Let C be a vertex cover of G with size at most k. For $v \in C$, add the edges v_q (for $1 \leq q \leq n-1$) to the matching M. For $u \notin C$, add the edges $\bar{v}_q$ (for $1 \leq q \leq n$) to the matching. This would result in a matching of size at least $k \cdot (n-1) + (n-k) \cdot n = n^2 - k$. All of the edges in M are non-adjacent, thus it is a valid matching. A constraint $\{u_i, v_j\}$ in H means that there is an edge $\{u, v\}$ in G. Since C is a vertex cover, either u or v is in C. By construction of the matching, where for a vertex in C all of its positive edges were added into the matching, this means that either u_i or v_j are in the matching. Therefore, all constraints in H are satisfied, and M is consequently a covering matching of (G', H').

($\Leftarrow$) Assume M is a covering matching of (G', H') of size at least $n^2 - k$. Then there must be at least $n-k$ vertices which have all negative edges in the matching. Otherwise, the matching has size at most $(n-k-1) \cdot n + (k+1) \cdot (n-1) = n^2 - k - 1 < n^2 - k$. For all the vertices which do not have all negative edges matched (at most k of those), add them to the set C. Then, for a vertex $v \notin C$, by definition of C all of its negative edges must be in M. By definition of constraints, for an edge $\{v, w\}$ in G, one positive edge of v must be in a constraint with one positive edge of w. Let the corresponding edges be denoted as v_p and w_p. As all of the negative edges of v are in M, due to how the graph was constructed, v_p is not in M. As a consequence, for the constraint $\{v_p, w_p\}$ to be satisfied, w_p must be in M. Since a positive edge is always adjacent to a negative one, this means that at least one negative edge of w is not in M. For a vertex to be in C, there has to be at least one negative edge missing, and so w is in C and the edge $\{v, w\}$ is covered by C. Therefore, C is a vertex cover of G. □

Now we use the previous reduction in order to show that, when the graph is either a path or a cycle and each edge participates in at most one constraint, the problem still remains NP-hard. A similar procedure can be performed for the CF-MM.

The idea of the proof is to iteratively merge two paths into a single path while changing the size of the optimal solution only by a constant. So in order to construct a path, $n-1$ merges are required, and to yield a cycle one merges the final with

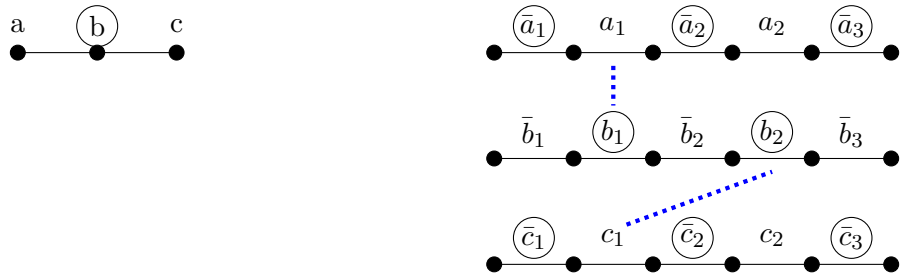


Figure 3.4. An example of the reduction in Lemma 3.4 The black lines indicate the edges, and the blue dashed lines represent the constraints. The vertices/edges are part of the vertex cover/matching. One sees that the size of the matching is $n^2 - 1 = 8$

itself.

Lemma 3.5. Cov-MM-$(\mathcal{P}, \Delta_1)$ and Cov-MM-$(\mathcal{C}, \Delta_1)$ are NP-hard.

Proof. We use the instance resuling from the reduction in the proof of Lemma 3.4. As shown in Figure 3.4, the two leaf edges of each path are not in a forcing constraint, and thus one can merge two such paths into one, with the new path having two of its leaf edges not in a constraint as well. This invariant holds only if the length of paths is at least 3, namely if $|V_G| \geq 2$, which can be assumed without loss of generality. Assume the two paths to be merged are $(\bar{a}_n, ..., \bar{a}_1)$ and $(\bar{b}_1, ..., \bar{b}_n)$, and the two paths are to be merged at $\bar{a}_1$ and $\bar{b}_1$. Connect the two paths via 5 new edges e_i for $1 \leq i \leq 5$, yielding an induced path $(\bar{a}_n, ..., \bar{a}_1, e_1, ..., e_5, \bar{b}_1, ..., \bar{b}_1)$. Add two new forcing constraints $\{\bar{a}_1, e_2\}$ and $\{\bar{b}_1, e_4\}$. Then for $0 \leq k \leq E(G)$:

There exists a covering matching in (G, H) of size at least $k \iff$ There exists a covering matching of (G', H') of size at least $k + 2$

($\Rightarrow$): Let M be a covering matching of (G, H) of size at least k. Then let $M' := M \cup \{e_2, e_4\}$. It is always possible to add these two edges, and satisfy the newly introduced constraints regardless of whether $\bar{a}_1$ or $\bar{b}_1$ are in M. The size of M' is also at least $k + 2$.
($\Leftarrow$): Let M' be a covering matching of (G', H') of size at least $k + 2$. Then let $M := M' \cap E_G$. Due to the new constraints $\{\bar{a}_1, e_2\}$ and $\{\bar{b}_1, e_4\}$, it is ensured that neither e_1 nor e_5 are in M'. Therefore, it is never possible for 3 edges out of $\{e_1, ..., e_5\}$ to be in M', and thus M is going to have size at least k.

One can perform $n - 1$ such path merges to yield a single path. For a cycle, one merges an additional time the final path with itself. An important invariant to be maintained during merging is for the new paths to not have the leaf edges in a constraint, which the reduction correctly guarantees.

With each path merge, there are 4 new vertices, and thus the size of the final graph is going to be $5|V_G| - 4$ for a path and $5|V_G|$ for a cycle. □

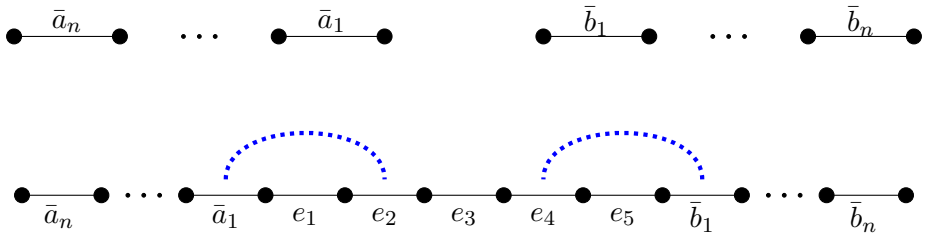


Figure 3.5. An example of a path merge shown in proof of Lemma 3.5. The reudction merges paths $(\bar{a}_n, ..., \bar{a}_1)$ and $(\bar{b}_1, ..., \bar{a}_n)$ at $\bar{a}_1$ and $\bar{b}_1$. The top shows the two paths before the merge, the bottom path after the merge, with the blue dotted lines indicating the newly added constraints.

Unfortunately, restricting the forcing graph to a regular one also does not make the problem polynomially solvable. In fact, it is only possible for 0-regular forcing graphs, which is equivalent to the Maximum Matching problem.

We have shown that the problem is NP-hard on graphs with maximum degree 1. The idea is that isolated edges in G can always be added to the final solution, regardless of whether they are in a constraint or not. We can use that to keep extending the two input graphs to achieve the desired degree for all vertices.

For maximum degree d of H, to achieve a desired maximum degree of $k \geq d \geq 1$ one adds $k+1$ isolated edges into G and makes them a clique in the forcing graph. Since we added $k+1$ isolated edges, the optimum solution size is increased by $k+1$.

To get a k-regular H, we could use the result obtained by Erdős and Kelly [8], which shows that every graph Γ with n vertices can be extended with at most n vertices to a graph Γ' with $\Gamma' \in \mathcal{R}_{\Delta_\Gamma}$ and $\Gamma = \Gamma'[V_\Gamma]$, meaning Γ is isomorphic to a subgraph of Γ'. In our case, because we can achieve the maximum degree of k in Γ, the graph Γ' graph would have regular degree k.

Before we proceed, we prove the following claim:

Claim 3.1. For a feasible instance (G, H) of Cov-MM, adding $k \geq 0$ isolated edges to G increases the size of the maximum covering matching by k.

It is important to note that there are no restrictions on whether the new edges are in a constraint; it is irrelevant for the proof and will be useful later.

Proof. Let (G', H') be the graphs obtained by adding k isolated edges to G in an instance (G, H). Since we do not introduce any constraints between the edges already present in G, we know that H is an induced subgraph in H'. Let t and t' be the sizes of the maximum covering matchings of (G, H) and (G', H'), M and M' the respective covering matchings of the largest size, and N the set of new isolated edges added to G. Then $M \cup N$ is a valid covering matching in (G', H'). Firstly, $M \cup N$ is a matching in G', because the edges in N are not adjacent to any other edges. It is also covering, because due to H being an induced subgraph of H', M satisfy all constraints within $H'[V_H] = H$ and N satisfy all constraints between V_H (old edges) and N, as well as witjin N. Therefore, $t' \geq t + k$.

Now we show the other direction. Since N is of size k, there must be at least $t' - k$ edges of the matching in $H'[V_H] = H$. Because H is an induced subgraph of H', the matching $\bar{M} := M \cap E_H$ satisfies all contraints in H, which makes it a valid covering matching, and therefore $t \geq t' - k$, or equivalently, $t' \leq t + k$.

From the two inequalities we conclude that $t' = t + k$. □

Now all the tools needed to prove Lemma 3.6 are at our disposal. To this end, we reduce from an instance of Cov-MM-$(\mathcal{G}, \Delta_1)$, which was shown to be NP-hard in Lemma 3.4. An example of the reduction is shown in Figure 3.6.

Lemma 3.6. Cov-MM-$(\mathcal{G}, \mathcal{R}_d)$ for $d \in \mathbb{N}$ is NP-hard.

Proof. Let (G, H) be an instance for Cov-MM-$(\mathcal{G}, \Delta_1)$, and k the size of its maximum covering matching. Assume without loss of generality that (G, H) is feasible (due to Lemma 3.7 to be proven next).

Let $d \in \mathbb{N}$ be the desired degree of the forcing graph. Initially, we make the instance have a maximum degree of d. For this we introduce $d + 1$ isolated edges to G, and make them a clique in the forcing graph. This results in maximum degree of the forcing graph of d (since $d \geq 1$ and H has max degree of 1). Let (G_1, H_1) be the resulting instance and k_1 the size of its maximum covering matching. Then, G_1 has $|V_G| + 2d + 2$ vertices (2 for each new isolated edge), H_1 has $|V_H| + d + 1$ vertices, and by Claim 3.1, $k_1 = k + d + 1$.

The next step is to reduce (G_1, H_1) to an instance with a d-regular forcing graph. To this end, we use the result obtained by Erdős and Kelly [8], which states that one can always extend a graph K with max-degree d_k with at most $|V_K|$ new vertices to a d_k-regular graph, in which K is an induced subgraph. We can therefore extend H_1 to a d-regular graph with at most $|V_{H_1}|$ new vertices. Let N be the set of those new vertices. Then, to represent the new vertices in G, we add $|N|$ new isolated edges to G_1.

Let (G_2, H_2) be the resulting instance and k_2 the size of its maximum covering matching. Then G_2 has $|V_{G_1}|+2|N|$ vertices (2 for each new edges), H_2 has $|V_{H_1}|+|N|$ vertices, and by Claim 3.1, $k_2 = k_1 + |N| = k + d + |N| + 1$.

As a consequence, we have proven the following equivalence:

(G, H) has a covering matching of size $k \iff (G_2, H_2)$ has a covering matching of size $k + d + |N| + 1$ □

Although an instance for Cov-MM does not necessarily have a covering matching of any size, checking whether that is the case is relatively easy.

Lemma 3.7. Given an instance (G, H) for Cov-MM, deciding whether there exists a covering matching in (G, H) can be verified in polynomial time.

Proof. We show this by reducing the input graphs to an instance of 2SAT, whose satisfiability will be equivalent to the feasibility of the input instance. Since 2SAT satisfiability is a problem solvable in polynomial time, the result follows. Let (G, H) be an instance of Cov-MM. Construct a 2SAT instance Φ with a variable set E_G. Two positive literals e_1 and e_2 are in a clauseif there exists an edge $\{e_1, e_2\} \in E_H$, and two negated literals $\bar{e}_1$ and $\bar{e}_2$ ($\bar{e}_1 \neq \bar{e}_2$) are in a clause they are adjacent in G. Then

There exists a valid covering matching in (G, H) $\iff$ Φ is satisfiable

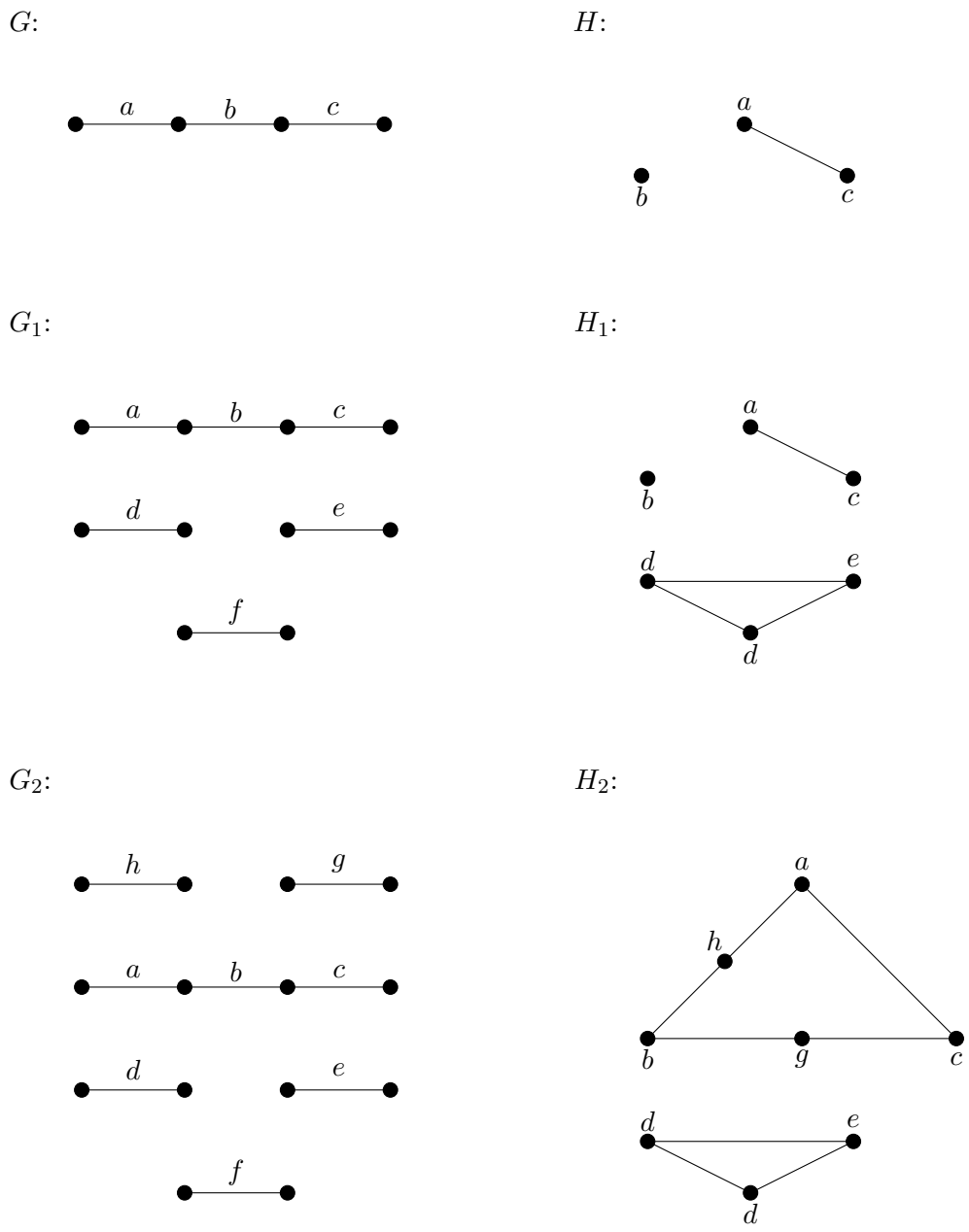
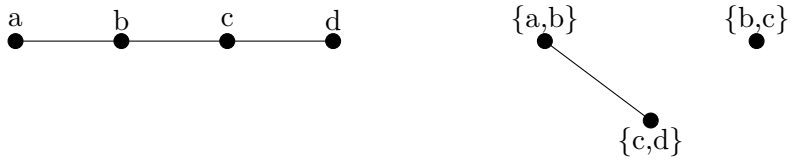


Figure 3.6. The 2 steps of the reduction shown in Lemma 3.6, with goal regular degree of 2. First the 3 isolated edges d, e, f are added to G to make the graph have max-degree 2. Later, the forcing graph is turned into a 2-regular graph by adding new edges h, g to G and introducing the appropriate constraints. To keep the illustrations clear, the constraints are not represented in G, G_1, and G_2. As can be checked, the only maximum covering matching of (G, H) is $\{a, c\}$. The maximum covering matching in (G_2, H_2) is $\{a, c, d, e, f, g, h\}$, which includes all of the new edges created during the process.



$$(\overline{\{a,b\}} \vee \overline{\{b,c\}}) \wedge (\overline{\{b,c\}} \vee \overline{\{c,d\}}) \wedge (\{a,b\} \vee \{c,d\})$$

Figure 3.7. An example of the reduction of Lemma 3.7. G top left, H top right, and Φ at the bottom. A matching $\{\{a,b\},\{c,d\}\}$ is a valid covering matching in (G,H). By setting $\{a,b\}$ and $\{c,d\}$ to true, and $\{b,c\}$ to false one gets a satisfying assignment for Φ.

($\Rightarrow$): Let M be a valid matching for (G,H). Then set the variables for edges in M to true, and others to false. By construction, all the clauses with only positive literals represent the constraints, and thus are satisfied by the assignment. For the clauses with only negated literals, because M being a matching, no two adjacent edges are in M. The variables of these edges therefore are not assigned to true simultaneously, and thus the clauses are satisfied.
($\Leftarrow$): Let S be the satisfying assignment for Φ and T the variables which are set to true by S. Then take all the edges of variables in T into the matching M. For two adjacent edges $e_1, e_2 \in E_G$, there exists a clause $(\bar{e}_1 \vee \bar{e}_2)$ in Φ. Thus, at least one of those edges is not in M, and so no two edges in M are adjacent. For a forcing constraint $\{e_i, e_j\}$ there exists a clause $(e_i \vee e_j)$, and so at least one of the two is in T, and therefore in M. Hence, M is a valid covering matching of (G,H). □

The ability to model the Cov-MM instance via a 2SAT formula can help us finding a valid covering matching. Because any extension of a vertex cover is still a vertex cover, we can extend a valid covering matching to a maximal matching. Since every maximal matching is at least half the size of the maximum matching, we have the following result.

Corollary 3.1. There is a 2-approximation algorithm for Cov-MM.

Proof. The previous reduction to 2SAT from Lemma 3.7 can be used to find a valid solution S for the Cov-MM instance. Since a satisfying assignment for a 2SAT formula can be found in polynomial time, we also get a valid covering matching in polynomial time. Let M be that matching. We first remove all edges in G and their respective vertices in H which are adjacent to some edge in M, since they cannot be simultaneously with M in a valid matching. In the remaining graph we greedily choose edges until it is no longer possible and add them to M. Let M' be the resulting matching. Then M' is a maximal matching of G, because otherwise it means that an edge was not added when greedily picking edges, which cannot occur. Because the size of any maximal matching is at least half of the size of maximum matching, and thus at least half of the size of the maximum covering matching (a covering matching cannot be larger than the maximum matching), we get a polynomial 2-approximation algorithm. □

It turns out that with some additional restrictions on the input graphs one can make the Cov-MM problem easy enough to be able to solve it in polynomial time. Next we present one such case. As a reminder, $M(K)$ denotes the size of the maximum matching (without constraints) of a graph K.

Lemma 3.8. Cov-MM with $M(G) \leq M(H)$ is in P.

The idea is that, when $M(H)$ is larger than the maximum possible matching, then the instance is infeasible, and when $M(G) = M(H)$, every valid matching is an optimal one.

Proof. The size of the maximum matching gives a lower bound on the size of the vertex cover. Therefore, the size of a vertex cover of H is at least $M(H)$. Because a valid covering matching of (G, H) is a vertex cover in H, a valid covering matching is of size at least $M(H)$. We proceed next by case distinction on $M(G)$.

Case 1. $M(G) < M(H)$. A valid covering matching of (G, H) is also a matching in G, and thus its size cannot exceed $M(G)$, and so has to be of smaller size that $M(G)$. But due to previous arguments, its size also has to be at least $M(H)$, a contradiction. Therefore in this case the instance is infeasible.

Case 2. $M(G) = M(H)$ Since the feasibility can be decided in polynomial time, without loss of generality, the instance (G, H) is feasible. Let M be a covering matching in (G, H) and k its size. Using the same argumentation as in the previous case, $k \geq M(H) = M(G)$ and $k \leq M(G)$, and so k is exactly $M(G)$. Any valid covering matching for (G, H) therefore has cardinality $M(G)$, and so any valid solution of (G, H) is an optimal one. Such a solution can be extracted from a satisfying assignment for the 2SAT formula of the Cov-MM instance introduced in Lemma 3.7. □

Due to restrictions on the forcing graph shown previously, we can also bound the number of edges in H, which is the number of constraints, from above.

Corollary 3.2. Let (G, H) be a feasible instance for Cov-MM. Then $|E_H| \leq \frac{n(n-1)(n+2)}{4}$.

Proof. Since the instance is feasible, it follows that $M(H) \leq M(G) \leq \frac{n}{2}$ as a consequence of Lemma 3.8. Let M be maximum matching of H, $V(M)$ the vertices that are matched in M, and $I := V \backslash V(M)$ the vertices which are not matched. Due to M being maximal, I is an independent set. Otherwise, we could extend M with an edge, which contradicts its optimality. Therefore there can only be edges within $V(M)$ and between $V(M)$ and I.

Because $|M| \leq \frac{n}{2}$, we have $V(M) \leq n$, and thus the number of edges within $V(M)$ is at most $\frac{n(n-1)}{2}$. For every edge $\{e_1, e_2\} \in M$, there cannot exist two distinct vertices e_1', e_2' in H which are not matched by M (and are therefore in I), such that e_1' is a neighbour of e_1 and e_2' is a neighbour of e_2, otherwise the matching $M := (M - \{\{e_1, e_2\}\}) \cup \{\{e_1, e_1'\}, \{e_2, e_2'\}\}$ has larger cardinality than M, a contradiction. Hence, either both of the endpoints of an edge in the matching have at most one common neighbour in I, or one endpoint is connected to multiple vertices in I and the other therefore is not. The latter is the worse case, and so the

number of edges between I and $V(M)$ is at most $|I| \cdot |M| \leq \frac{n(n-1)}{2} \cdot \frac{n}{2} = \frac{n^2(n-1)}{4}$, where $|I| \leq \frac{n(n-1)}{2}$ because I is a subset of V_H and $V_H = E_G$.

Together with at most $\frac{n(n-1)}{2}$ edges within $V(M)$ it yields an upper bound of $\frac{n(n-1)(n+2)}{4}$. □

The next result outlines the other side of the coin: there is also a bound on the number of constraints from below, starting from which both variants are NP-hard.

Lemma 3.9. CF-MM/Cov-MM with $|E_H| \in \Omega(|V_H|^{1-\varepsilon})$ is NP-hard for any $1 > \varepsilon > 0$.

Proof. Without loss of generality we assume that $|E_H| \geq 1$ and proceed via a reduction from CF-MM/Cov-MM with $|E_H| = \frac{|V_H|}{2}$, which is NP-hard (Theorem 12 [1], Lemma 3.6). We add $\lceil |E_H|^{\frac{1}{1-\varepsilon}} \rceil$ many new independent edges to G, which eventually are isolated vertices in the forcing graph. We call the new graphs G' and H'. Then it holds that $|V_{H'}| = |V_H| + \lceil |E_H|^{\frac{1}{1-\varepsilon}} \rceil = 2|E_H| + \lceil |E_H|^{\frac{1}{1-\varepsilon}} \rceil \leq 4|E_H|^{\frac{1}{1-\varepsilon}}$. The number of edges in H' however stays the same (only isolated vertices are added), namely $|E_{H'}| = |E_H|$. Therefore, $|E_{H'}| = |E_H| = \left(|E_H|^{\frac{1}{1-\varepsilon}}\right)^{1-\varepsilon} \geq (\frac{|V_{H'}|}{4})^{1-\varepsilon}$, which is within the specified bounds.

Because only independent edges were added to the graph G, and are not constrained, the size of the maximum conflict-free/covering matching is increased exactly by the number of new edges. Therefore, a solution for the new instance with the newly added edges removed gives the solution for the initial one and vice versa. □

Chapter 4

Parameterized Complexity

Parameterized complexity is not only useful theory outlining the possible hierarchy within the NP class, but also has benefits in practice. Sometimes the problem is NP-hard, but most of the cases which occur in practice deviate from the hardest problem instances and possibly have some bounded structural parameter. This is where parameterized complexity of the problem comes in handy, because an FPT algorithm would eventually yield a polynomial time algorithm for the majority of the instances in the real world. In this thesis, we consider the solution size, maximum matching, treewidth, and maximum degree as parameters.

The solution size as a parameter for CF-MM was extensively studied by Agrawal et al. [2], with restrictions to various graph classes. They have shown that the problem is W[1]-hard when parameterized by the solution size. We additionally show that CF-MM is in FPT when parameterized by the solution size together with the maximum degree of the conflict graph by reducing it to the d-Set-Packing problem. For the covering variant, we show a simple algorithm parameterized by the solution size which involves branching on one constraint in each step. We further consider the maximum matching of G, and show that the problem is W[1]-hard for CF-MM, and in FPT for Cov-MM.

We furthermore what the previous results in this thesis imply about the complexity of the treewidth and maximum degree as parameters for both problems. We additionally introduce the extended constraint graph, and show that both variants are in FPT when parameterized by its treewidth. We also show how the new graph can be used to find an approximation algorithm for CF-MM in terms of the maximum degrees of G and H, and generalize the previous known result to an $(\Delta_H + 1)$-approximation algorithm for the weighted CF-MM.

We outline results related to this chapter in Table 4.1. Our results are in blue.

4.1 Solution size as a parameter

We start with the most straightforward parameter, and already here one can see the disparity of complexity between the two variants. While CF-MM is W[1]-hard, the covering version is in FPT. We proceed by first outlining the result related to CF-MM, which implies inapproximability on the side.

Problem / Parameter	k	$k+d$	$tw(G)$	$tw(H)$	Δ_H	$k + \Delta_H$	$tw(F)$	$M(G)$
CF-MM	W[1]-hard [2]	-	PARA-NP-hard	PARA-NP-hard	PARA-NP-hard	FPT	FPT	W[1]-hard
CF-MM with chordal H	FPT [2]	-	?	?	?	FPT, implied by above	FPT, implied by above	?
CF-MM with d-degenerate H	?	FPT [2]	?	?	?	?	?	?
Cov-MM	FPT	-	para-NP-hard	para-NP-hard	para-NP-hard	-	FPT	FPT
Weighted Cov-MM	?	-	para-NP-hard	para-NP-hard	para-NP-hard	FPT	FPT	FPT

Table 4.1. Summary of the results related to parameterized complexity of the two problems. F here denotes the extended constraint graph, which is introduced later. Our results are in blue.

Lemma 4.1 (Theorem 3, [2]). CF-MM is W[1]-hard when parameterized by the solution size.

The proof proceeds by reducing from the Independent Set problem, which is W[1]-hard when parameterized by the solution size [3]. We outline the proof here for completeness.

Proof. Let G be an instance for the Independent Set problem. We construct an instance (G', H') for CF-MM as follows. For every $v \in V_G$ we add an edge $e_v := \{v, v'\}$ to G'. For an edge $\{u, v\} \in E_G$ we add an edge $\{e_u, e_v\}$ to H'. The construction ensures that H' is isomorphic to G.

Since there are no adjacent edges in G, any independent set of G is a valid conflict-free matching in (G', H'). Additionally, any conflict-free matching of (G', H') is an independet set in H', which is isomorphic to G and therefore also an independent set in G. □

One can additionally question whether the problem is also W[2]-hard when parameterized by the solution size, however, this is very unlikely given that there is parameterized reduction to Independent Set, which is outlined later when considering the *Extended constraint graph* in Lemma 4.8. Were the problem W[2]-hard, a parameterized reduction from the Dominating Set to the Independent Set problem would follow, implying that W[1]=W[2].

The Independent Set problem is NP-hard to approximate within a factor of $n^{1-\varepsilon}$ for any $\varepsilon > 0$. According to Theorem 10.9 ([9]), approximating maximum

clique in polynomial time within a factor of $n^{1-\varepsilon}$ for any $\varepsilon > 0$ is not possible, unless P = NP. Because an independent set in a graph is a clique in its complement, an $n^{1-\varepsilon}$ approximation algorithm would yield an approximation for the Maximum Clique problem with the same factor.

Since there is a one-to-one correspondence between the maximum independent set of the input graph and the maximum conflict-free of the reduced instance in the proof of Lemma 4.1, this implies the following approximation lower bound for the general CF-MM.

Corollary 4.1. There is no polynomial-time $(n^{1-\varepsilon})$-approximation algorithm for the CF-MM problem for any $\varepsilon > 0$, unless P = NP.

The solution size as a parameter, however, is not completely hopeless for the general CF-MM problem, and can help to get a parameterized algorithm when paired up with the maximum degree of the conflict graph.

Lemma 4.2. CF-MM is FPT when parameterized by $k + \Delta_H$, where k is the output solution size and Δ_H is the maximum degree of the conflict graph.

We show this via a parameterized reduction to d-Set-Packing, which is defined as follows:

d-Set-Packing	
Given:	A universe $\mathcal{U}$, a family $\mathcal{S}$ of sets over $\mathcal{U}$, where each set in $\mathcal{S}$ is of size at most d, and an integer k'.
Question:	Does there exist a family $\mathcal{S}' \subseteq \mathcal{S}$ of k' pairwise disjoint sets?

It admits a kernel with at most $\binom{k'd}{d}$ sets and at most $\binom{k'd}{d} \cdot d$ elements, where k' is the output solution size, and d is the size of the largest set in the family $\mathcal{S}$ of the input instance (Theorem 12.20, [3]). A kernel parameterized by $(k' + d)$ is equivalent to the existence of an FPT algorithm according to the same parameter. Therefore, d-Set-Packing is also FPT when parameterized by $k' + d$. Therefore, to show that CF-MM is in FPT when parameterized by $k + \Delta_H$, a parameterized reduction to d-Set-Packing is sufficient.

During the reduction, we construct an instance for d-Set-Packing, such that every edge of the original graph is encoded in a set, and two sets whose edges are in conflict with one another have at least one common element, so that both of the sets corresponding to the conflicting edges cannot be in the valid solution.

Proof. Let (G, H) be an instance for the CF-MM problem, and k the requested minimum solution size. We construct an instance for d-Set-Packing denoted as $(\mathcal{U}, \mathcal{S})$ as follows.

- $\mathcal{U}$, the universe of elements, contains all vertices and constraints as elements. Formally, $\mathcal{U} := \{v \mid v \in V_G\} \bigcup \{e_{ij} \mid e_{ij} \in E_H\}$.
- The family of sets $\mathcal{S}$ contains for each edge $i := \{u, v\} \in E_G$ a set C_i, defined as $C_i := \{u, v\} \bigcup \{e_{ij} \mid e_{ij} \in E_H\}$. We do not consider multigraphs in this work, so C_i's uniquely identify the edges.

What is left to show is that:

There exists a conflict-free matching in (G, H) of size at least k
$\iff$
There exists a set packing $S' \subseteq S$ of $(\mathcal{U}, \mathcal{S})$ of size at least k

($\Rightarrow$) : Let M be a conflict-free matching in (G, H) of size at least k. Then construct a packing S', which includes the sets corresponding to the edges in M. Formally, $S' := \{C_i \mid e_i \in M\}$. Then S' is of size at least k, and the sets in S' are pairwise disjoint. This is because otherwise there must exist a pair $C_i, C_j \in S'$ with $C_i \cap C_j \neq \emptyset$. Let x be some element from $C_i \cap C_j$. Then either x is an element representing a vertex of G or a constraint in H.

If x represents a vertex, by definition of C's that implies that M hast at least 2 adjacent edges, which is a contradiction, since M is a matching. If x represents a constraint that means that the respective edges in G are in conflict with each other in H, which is again a contradiction, since M is conflict-free. Therefore, the sets in S' are indeed pairwise disjoint.

($\Leftarrow$) : Let S' be a set packing of $(\mathcal{U}, \mathcal{S})$ of size at least k. Then set $M := \{e_i \mid C_i \in S'\}$, which is of size at least k, since C_i's uniquely identify the edges.

Firstly, M is a matching, because if two adjacent edges $i := \{u, v\}$ and $j := \{u, w\}$ for some $u, v, w \in V_G$ would have been in M, then the sets C_i and C_j, both of which are in S' by construction of M, would have had an element u in common, which is a contradiction to S' being a set packing.

M is also conflict-free, because otherwise if some edges i and j in M are in conflict with each other, that implies that C_i and C_j (which are in S') have a common element e_{ij}, which is a contradiction to S' being a valid set packing.

Since the FPT algorithm for d-Set-Packing is parameterized by $(k' + d)$, where k' is the requestion solution size and d is the size of the largest set in the family $\mathcal{S}$ of the input instance $(\mathcal{U}, \mathcal{S})$, what is left to be done is to show that $(k' + d)$ of the reduced instance can be upper bounded by some computable function $f(k, \Delta_h)$.

In the reduction, the requested solution size remains the same, thus $k' \leq k$. The size of each set C_i in $\mathcal{S}$ depends on the number of contstraints that an edge i is a part of in (G, H). The set C_i has 2 elements for each endpoint of the corresponding edge i, as well as one element for each conflict the edge is part of. Therefore the set of C_i for any $i \in E_G$ is at most $\Delta_H + 2$, so the maximum size of any set in $\mathcal{S}$ (denoted as d) is at most $\Delta_H + 2$.

Therefore, $k' + d \leq k + \Delta_H + 2$. □

For the Cov-MM problem however, a simple technique is enough to get an exponential-time algorithm with respect to the solution size.

The procedure is a branching algorithm, where in a single step one constraint is considered. Because for every constraint at least one of its edges has to be in the final solution, the parameter is decreased by at least one with each recursion step. This leads to a recursion tree of depth at most k, from which the runtime follows.

We briefly describe the behaviour of Cov-MM-TakeEdge and Cov-MM-Sol, and then proceed to prove their correctness and runtime.

We first describe Cov-MM-TakeEdge as it is used as a subroutine in Cov-MM-Sol. It takes as input 4 parameters: graphs G and H which identify the instance for Cov-MM, solution size k, and e which is an edge in G. The output of the algorithm is, in simple terms, an instance (G', H') and solution size k' that are a result of including the edge e into a covering matching. For the sake of simplicity, we assume that the algorithm is only going to be called on feasible instances (disregarding the solution size), and the algorithm therefore skips the feasibility check.

In order to achieve this goal, there are 2 things that need to be done: the edges which are adjacent to G need to be removed and subsequently the edges in a constraint with those edges need to be included into the solution. Since edges have to be continously added into the solution, we keep looping until we have removed/included all edges required. This is all done in a while loop with a queue. Throughout the whole phase the algorithm also keeps track of how many edges were included, and decreases the solution size parameter k each time.

After having done the previously described steps, the algorithm then returns the instance only with the edges which are neither definitely included in nor excluded from the solution. The algorithm also returns the modified solution size parameter k.

Next we consider Cov-MM-Sol. Initially the algorithm gets rid of infeasible instances which is done in two steps: using the upper bound on the constraints from Corollary 3.2 and reducing to a 2SAT formula like in Lemma 3.7. If the input instance does not satisfy both of these conditions, we safely return false.

If the instance has no constraints, the problem boils down to deciding whether there is a matching of size at least k, and the ouput is respectively that decision. If $k = 0$, then the question is whether there is a covering matching of any size, and since the feasibility check has been done beforehand, we return true.

If there is at least one constraint, an arbitrary one denoted as $\{e_1, e_2\}$ is taken. Then for edge e_1 the algorithm checks whether the instance remains feasible if it is taken into the solution. If yes, then (G_1, H_1, k_1) is assigned an instance which is a result of including e_1 into a solution. To get such an instance, the algorithm calls the Cov-MM-TakeEdge procedure. If e_1 cannot be in the final solution, (G_1, H_1, k_1) is assigned a trivial No-instance. The same procedure is done for e_2. The decision of the algorithm is then whether one of the instances created is a Yes-instance.

At this stage we show that the algorithms outlined are also correct and run in the desired runtime. We start again with Cov-MM-TakeEdge, and proceed later with Cov-MM-Sol.

Lemma 4.3. Under the assumption that (G, H) has a valid covering matching with edge $e \in E_G$ in it, Cov-MM-TakeEdge(G, H, k, e) (Algorithm 2) is correct and runs in time $\mathcal{O}(n^3)$.

Proof. Initially the algorithm creates a new queue and pushes e into it. The queue itself is going to be used to only keep edges which are to be included into a covering matching as a result of including e. Arrays *in* and *out* are initialized with size $|E_G|$ to False, and indicate whether the corresponding edge is to be included in or excluded from the matching respectively. Since e is included, $in[e]$ is correctly set to True.

Further while loop serves the purpose of including and excluding all edges required, as well as keeping track of the solution size k. Each element in the queue is to be included into the solution, and so each iteration is going to pop an edge e_{curr} from

Algorithm 1 Cov-MM-Sol(G, H, k)

```
1: if |E_H| > n(n−1)(n+2)/4 then   // |E_H| is the number of constraints
2:     return False
3: Φ ← Cov-MM-2SAT(G, H)
4: if not 2SAT-Check(Φ) then
5:     return False
6: if |E_H| = 0 then   // No constraints
7:     if M(G) ≥ k then
8:         return True
9:     else
10:        return False
11: if k = 0 then
12:    return True
13: {e_1, e_2} ← edge from E_H
14: if 2SAT-Check(Φ[e_1 = 1]) then
15:    (G_1, H_1, k_1) ← Cov-MM-TakeEdge(G, H, k, e_1)
16: else
17:    (G_1, H_1, k_1) ← ((∅, ∅), (∅, ∅), 1)   // A trivial No-instance
18: if 2SAT-Check(Φ[e_2 = 1]) then
19:    (G_2, H_2, k_2) ← Cov-MM-TakeEdge(G, H, k, e_2)
20: else
21:    (G_2, H_2, k_2) ← ((∅, ∅), (∅, ∅), 1)
22: return Cov-MM-Sol(G_1, H_1, k_1) ∨ Cov-MM-Sol(G_2, H_2, k_2)
```

the queue and include it. For this the solution size parameter needs to be decreased, and in order not to have negative value as a parameter, the maximum of $k-1$ and 0 is taken. Since a matching is required and e_{curr} is to be included, all of its adjacent edges in G can no longer be included. All of those edges are in the list adj. For any element e_{rem} is adj, its entry has to be set to True in the out array. Since the edge e_{rem} is excluded from the solution, all of the edges it is in a constraint with have to be included (otherwise we don't have a valid covering matching). The algorithm therefore correctly loops through the open neighbourhood of e_{rem} and push each edge e_{add} into the queue. In order to avoid the same edge being pushed into the queue multiple times, as soon as an edge e_{add} is added, $in[e_{add}]$ is set to True, and everytime an element is to be added into the queue the algorithm verifies whether it has already pushed the considered edge into the queue.

Since the queue would never have an edge added into it more than once, it eventually terminates. An edge which is neither adjacent to one included into the matching, nor in a constraint with one excluded, can either be in the final matching or not. Therefore by running the while loop we exhaust all possibilities, and the edges e_cons for which neither $in[e_{cons}]$ nor $out[e_{cons}]$ are set to True stay in the instance. This is what lines from 16 to 21 are responsible for.

A scenario could occur when an edge is both to be included and excluded at the same time. However, since whenever we set in or out entry to True, we know for sure that the edge has to be definitely included/excluded. If such a case occurs, it implies that taking e into the covering matching is not possible, which is a contradiction.

Algorithm 2 Cov-MM-TakeEdge(G, H, k, e)

```
1: Q ← new Queue
2: Q.push(e)
3: in ← new Array(bool, |E_G|, False)
4: out ← new Array(bool, |E_G|, False) // Create 2 boolean arrays of size |E_G|
                                       to track edges which are either definitely in the
                                       or out of the matching
5: in[e] ← True
6: while not Q.isEmpty() do
7:     e_curr ← Q.pop()
8:     k ← max{0, k − 1}
9:     adj ← adjacent edges to e_curr
10:    for all e_rem in adj do
11:        out[e_rem] ← True
12:        for all e_add in N_H(e_rem) do
13:            if not in[e_add] then
14:                Q.push(e_add)
15:                in[e_add] ← True
16: E_stay ← ∅
17: for all e_cons in E_G do
18:    if (not in[e_cons]) ∧ (not out[e_cons]) then
19:        E_stay ← E_stay ∪ {e_cons}
20: G ← (V, E_stay)
21: H ← H[E_stay]
22: return (G, H, k)
```

Hence, we do not have to worry about such situations.

Now we proceed with the runtime. Initializing arrays in the beginning takes $\mathcal{O}(n^2)$. We have shown that the queue is only going to have edges which are included into a matching, and therefore there can be at most $M(G) \leq \frac{n}{2}$ elements ever appearing in Q, and therefore the number of iterations of the while loop is also bounded by $\frac{n}{2}$.

Constructing the adjacency list for the edge e_{curr} can be done via iterating through neighbours of the two of its endpoints, which is in $\mathcal{O}(n)$, which is also therefore the bound for the number of edges in adj. Since for an edge e_{rem} to be removed, all of the edges it is in a constraint with have to be included and the algorithm is correct, the degree of e_{rem} cannot exceed $M(G) \leq \frac{n}{2}$. Therefore, going through the open neighbourhood of e_{rem} is also done in $\mathcal{O}(n)$.

Since all of the loops have $\mathcal{O}(n)$ iterations, and there are 3 of them, the while loop runs in $\mathcal{O}(n^3)$ time.

Constructing E_{stay} is done in $\mathcal{O}(|E_G|)$, as lines 18 and 19 are done in constant time. Reconstructing G is done in $|V| + |E_{stay}| \in \mathcal{O}(n^2)$ time. Reconstructing H can be done via iterating through elements in E_H and only leaving constraints which have at least one of its endpoints in E_{stay}. As we assume that the algorithm is called only on feasible instances, $|E_H|$ is bounded from above by $\mathcal{O}(n^3)$.

All of the points mentioned above run in $\mathcal{O}(n^3)$, then so does Cov-MM-TakeEdge. □

Now we can proceed with the main algorithm.

Lemma 4.4. Cov-MM-Sol (Algorithm 1) solves Cov-MM in time $\mathcal{O}(2^k \cdot n^3 + n^4)$, where k is the requested solution size.

Proof. We start with the base cases. In order to decrease the initial runtime, the algorithm first checks whether the bound on the number of constraints holds, that is correct per Corollary 3.2. If it is higher, then the algorithm correctly returns false.

However, it is still possible that the instance is infeasible even when constraint upper bound holds. To get those instances out of the way the algorithm reduces the problem to a 2SAT formula Φ as outlined in the proof of Lemma 3.7. For the sake of simplicity, the function 2SAT-Check(Φ) stands for the algorithm which checks the feasibility of the input 2SAT formula Φ using the algorithm outlined by Aspvall et al. [10]. If the instance is infeasible, the algorithm correctly returns false. Important to note, the formula can only tell whether there is a valid covering matching for (G, H), not whether there is one of size at least k.

If $|E_H| = 0$, namely if there are no constraints, the question is essentially whether there exists a matching of size at least k. Therefore, the algorithm returns true only if $M(G)$ is at least k, and false otherwise.

If $k = 0$, the answer is equivalent to whether there is any covering matching. Since at this stage the algorithm only deals with feasible instances, it correctly returns true.

After having gone through base cases, the algorithm takes an arbitrary constraint from the instance. The correctness of the ensuing steps is based on the observation that, for a constraint $\{e_1, e_2\}$, all covering matchings (of any size) have either e_1 or e_2. Therefore, if we branch into the two cases, with each including either e_1 or e_2, we are able to search through all of the covering matchings. In order to have the branch instances to recurse on, the Cov-MM-TakeEdge algorithm is used, with the instances (G_1, H_1, k_1) for the branch with e_1 included, and (G_2, H_2, k_2) for the one with e_2. The algorithm then correctly returns the OR of the two branches.

The correctness of Cov-MM-TakeEdge holds under the assumption that including the given edge into a covering matching of the input instance does not make it infeasible. In order to verify whether it is possible to do so (disregarding the resulting solution size), the algorithm uses the equivalent 2SAT formula Φ. To model including a specific edge into the solution, all clauses with the positive literal of the edge are removed (since they are satisfied if the variable is set to true), and all occurrences of the negated literals are removed (as they cannot satisfy a clause). The resulting formula is shortened with $\Phi[e_i = 1]$, where e_i is the edge/variable which is set to true. Having done the modification, 2SAT-Check is called on the new formula, which is satisfiable if and only if the respective edge can be included in a covering matching. If the formula is satisfiable, then Cov-MM-TakeEdge can be called on the edge considered, and the branch instance is respectively assigned its output.

If it is not possible, then the algorithm assigns instances of $((\emptyset, \emptyset), (\emptyset, \emptyset), 1)$ in lines 17 and 21. It is a No-instance, as it asks for a matching of size at least one in an empty graph, and when recursing, Cov-MM-Sol correctly returns false in line 10.

Now we consider the runtime. The check of the number of constraints can be done in $\mathcal{O}(n^4)$, as there are at most $|E_G|^2 \leq n^4$ constraints. After the first line, the

algorithm only deals with instances of size in $\mathcal{O}(n^3)$.

The formula Φ consists of two types of clauses: the ones which encode that adjacent edge pairs in G cannot be simultaneously in a matching, and the ones which encode the covering constraints. The number of adjacent edge pairs in G is bounded by $\mathcal{O}(n^3)$, as for each of at most n^2 edges, there are at most $2n-2$ adjacent edges ($n-1$ for each of its endpoints). The number of constraints is also bounded by $\mathcal{O}(n^3)$ due to the check in line 1. Therefore, constructing Φ takes time $\mathcal{O}(n^3)$.

The 2SAT satisfiability algorithm presented by Aspvall et al. [10] runs in time $\mathcal{O}(v+c)$, where v is the number of variables and c is twice the number of clauses in the formula. Since each edge in G becomes a variable in Φ, the number of variables in Φ is at most n^2. As argumented above, the number of clauses is in $\mathcal{O}(n^3)$, and so 2SAT-Check(Φ) in line 4 runs in $\mathcal{O}(n^3)$.

It was shown by Vazirani et al. [11] that the cardinality of the maximum matching can be computed in $\mathcal{O}(|V|^{0.5} \cdot |E|)$ time. In our case, that would correspond to a runtime of $\mathcal{O}(n^{2.5})$ on line 7.

The new formulas $\Phi[e_1 = 1]$ and $\Phi[e_2 = 2]$ in lines 18 and 19 can be computed via iterating through all clauses, of which there are at most $\mathcal{O}(n^3)$. The resulting formulas are not larger than Φ, and so the time upper bound of $\mathcal{O}(n^3)$ in line 4 holds for them as well. Per Lemma 4.3, the calls in lines 15 and 19 run in $\mathcal{O}(n^3)$.

The only step with the $\mathcal{O}(n^4)$ runtime is the check of the number of constraints in 1. However, if the bound does not hold, no subsequent calls to Cov-MM-Sol follow. Therefore, $\mathcal{O}(n^4)$ is contributed only in the first invocation of the algorithm, and all the following ones contribute only $\mathcal{O}(n^3)$. Considering that the algorithm branches into two in each step, the total runtime of the algorithm is in $\mathcal{O}(2^h \cdot n^3 + n^4)$, where h is the height of the branching tree.

We argue that the height can be bounded by k. In case the recursion proceeds on the output of the Cov-MM-TakeEdge algorithm, then the solution size parameter is decreased by at least 1, as at least one edge is included into the matching. The only situation when that does not happen is in lines 17 and 21. However, the invocation on these two instances terminates in line 10 without any further recursive calls, therefore we can ignore such cases. The branching terminates without any recursion in case $k = 0$, which as argumented previously, is after at most k recursive calls. Hence, the height of the branching tree is bounded by k, which results in the desired runtime. □

A natural question would be to seek a $2^{o(k)} \cdot n^c$ algorithm for some $c \in \mathbb{N}$. However, modifying the reduction in Lemma 3.4 a little bit can be used to show that under ETH no such algorithm exists. In the next result, we construct a reduction in which a vertex cover of size at most k corresponds to a covering matching of size at least $2n-k$. Hence seeking a covering matching of size at least $2n-k$ for all $k \in [n]$ (without loss of generality, $k \neq 0$) and taking the maximum for which it is possible to do so would yield a $2^{o(n)} \cdot n^{c+1}$ algorithm for the Vertex Cover problem, which does not hold under ETH (Theorem 14.6, [3]). The aforementioned algorithm would exist because the total runtime of running reductions and the FPT algorithm for Cov-MM results in the following runtime:

$$\sum_{k=1}^{n} 2^{o(2n-k)} \cdot n^c \leq \sum_{k=1}^{n} 2^{o(n)} \cdot n^c \leq 2^{o(n)} \cdot n^{c+1}$$

Lemma 4.5. There is no $2^{o(k)} \cdot n^c$ algorithm for the Cov-MM problem, where $c \in \mathbb{N}$ and k is the requested solution size, unless ETH fails.

Now we actually show such a reduction, which is almost idential to the proof of Lemma 3.4. The difference here would be that instead of distributing the constraints among the edges of the path, we devote only one edge per path to this purpose. This allows us to reduce the length of each path to 3.

Proof. Let G be the input graph for the Vertex Cover problem with n vertices. We construct the input instance (G', H') for the Cov-MM problem in the following way. For every vertex $v \in V_G$, introduce a path of 3 edges e_i for $1 \leq i \leq 3$. We divide the edges into one positive ($e_v := e_2$) and two negative edges ($\bar{e}_{v1} := e_1,\ \bar{e}_{v2} := e_3$). For an edge $\{u, v\} \in E_G$, add a constraint $\{e_u, e_v\}$. Next we show the following:

For $k \leq n$:
There exists a vertex cover of G with size at most $k \iff$ There exists a covering matching M in (G', H') of size at least $2n - k$.

($\Rightarrow$) Let C be a vertex cover of G with size at most k. For $v \in C$, add the edge e_v into the matching M. For $u \notin C$, add the edges $\bar{e}_{v1}$ and $\bar{e}_{v2}$ into the matching. This would result in a matching of size at least $k + (n - k) \cdot 2 = 2n - k$. All of the edges in M are non-adjacent, thus it is a valid matching. A constraint $\{e_u, e_v\}$ in H means that there is an edge $\{u, v\}$ in G. Since C is a vertex cover, either u or v is in C. By construction of the matching, where for a vertex w in C its positive edge, namely e_w was added into the matching, this means that either e_w or e_v are in the matching. Hence, all constraints in H are satisfied, and M is consequently a covering matching of (G', H').

($\Leftarrow$) Assume M is a covering matching of (G', H') of size at least $2n - k$. Then there must be at least $n - k$ vertices which have two of their negative edges in the matching. Otherwise, the matching has size at most $(n-k-1)\cdot 2+(k+1) = 2n-k-1 < 2n-k$. For all the vertices which do not have both negative edges matched (at most k of those), add them to the set C. Assume for some non-isolated vertex v, it is not in C. By definition of C, both of the negative edges of v must be in M. By definition of constraints, for an edge $\{v, w\}$ in G, the edge e_v must be in a constraint with the edge e_w. As both of the negative edges of v are in M, and e_v is adjacent to both of them, it is not in M. As a consequence, for the constraint $\{e_v, e_w\}$ to be satisfied, e_w must be in M. Since a positive edge is adjacent to both negative ones, this means that none of the negative edges of w are in M. For a vertex to be in C, there has to be at least one negative edge missing, and so w is in C and the edge $\{v, w\}$ is covered by C. Therefore, C is a vertex cover of G. □

4.2 Maximum Matching as a parameter

Since the problems that this work deals with are generalizations of the Maximum Matching problem, one could potentially see additional information of the size of the maximum matching size as useful. For the covering variant it indeed is, and an FPT algorithm can be designed just by modifying Algorithm 1 to ignore the

requested solution size, and just branch as long as it is possible to do so. Since in each level the algorithm takes one edge into the matching, the height of the branch tree cannot exceed $M(G)$, yielding the desired runtime.

Lemma 4.6. Cov-MM is in FPT when parameterized by $M(G)$.

Just like in case of the solution size as the parameter, here the CF-MM also seems to be of much harder complexity than Cov-MM, and we show that it is W[1]-hard. This is also done via a reduction from the Independent Set problem parameterized by the solution size, like in the proof of Lemma 4.1.

Lemma 4.7. CF-MM is W[1]-hard when parameterized by $M(G)$.

Proof. Let G be the input for the Independent Set problem, and k the requested solution size. We construct an instance (G', H') for the CF-MM problem in the following way:

- G' has exactly k stars with $|V_G|$ edges, each representing a vertex in the original graph, namely
  - $V_{G'} := \{s_i \mid i \in [k]\} \bigcup \{v_i \mid i \in [k], v \in V_G\}$ and
  - $E_{G'} := \{\{s_i, v_i\} \mid i \in [k], v \in V_G\}$. Further on we use $e_{u,i}$ to denote an edge $\{s_i, u_i\}$.
- We introduce conflicts between the edges that correspond to the same vertex in the original graph, as well as conflicts between all edges representing u and v, in case they are neighbours in G. Formally,
  - $V_{H'} := E_{G'}$ and
  - $E_{H'} := \{\{e_{u,i}, e_{u,j}\} \mid i, j \in [k], i \neq j, u \in V_G\}$ $\bigcup\{\{e_{u,i}, e_{v,j}\} \mid i, j \in [k], \{u, v\} \in E_G\}$.

The construction ensures that $M(G) = k$, since exactly one edge from each star forms a maximum matching. There is an example of a reduction in Figure 4.1. Now we show the following:

For $k \leq n$:
There exists an independent set in G with size $k \iff$ There exists a conflict-free matching M in (G', H') of size k.

($\Rightarrow$) Assume G has an independent set I of size k. Then choose one corresponding edge for each vertex in I, namely for an arbitrary permutation I_p of vertices in I, a matching $M := \{e_{u,i} \mid i \in [k], u \text{ is the } i\text{-th vertex in } I_p\}$, which is of size k. It is a matching since at most one edge is taken from each star. It is also conflict-free because it does not violate any constraints: neither are there edges representing the same vertex simultaneously in the matching. nor are there conflicts between edges corresponding to different vertices, since they only are between neighbours in G and I is an independent set, so this case cannot occur as well.
($\Leftarrow$) Let M be a conflict-free matching in (G', H') of size k. One can include the respective vertex into the set for each edge in M. Namely, $I := \{v \mid e_{v,i} \in M\}$. It has

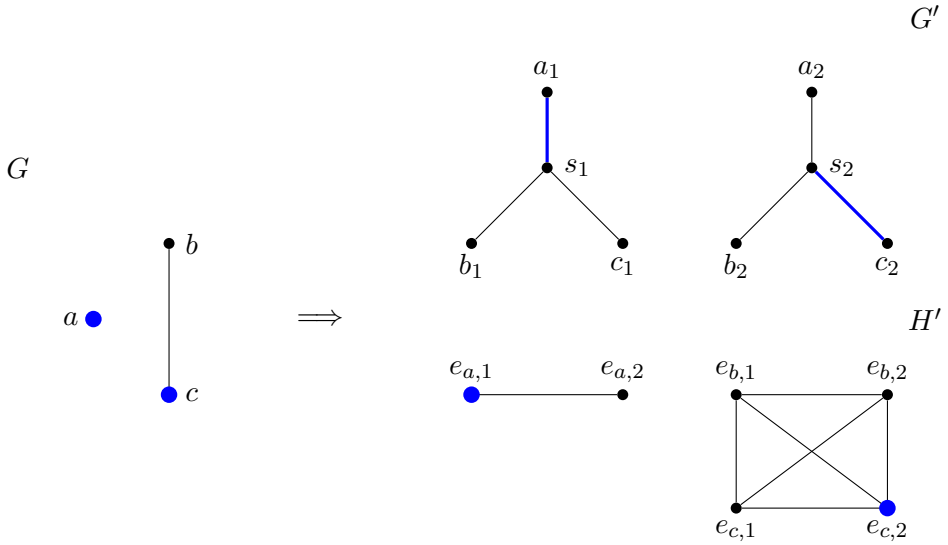


Figure 4.1. Example of the reduction in Lemma 4.7 with $k = 2$. An independent set $\{a, c\}$ can be translated into a matching $\{e_{a,1}, e_{c,2}\}$, which is outlined with blue nodes and edges.

size exactly k, because no two edges $e_{u,i}$ and $e_{u,j}$ for some i, j with $i \neq j$ can be in M due to the conflicts. Additionally, it is an independent set because otherwise for an edge $\{u, v\}$ in $G[I]$ the constraint $\{e_{u,i}, e_{v,j}\}$ for $e_{u,i}, e_{v,j} \in M$ would be violated. □

4.3 Treewidth as a parameter

The treewidth of either G or H is another parameter which makes the CF-MM problem W[1]-hard, however, compared with the solution size which allows for a parameterized algorithm for the Cov-MM problem, it makes the forcing variant W[1]-hard as well. In fact, for both variants, the treewidth of either G or H even for constant values does not allow for a parameterized algorithm, unless P = NP. This is the implication from the results outlined previously in this thesis.

The reductions in Lemma 3.1 and Lemma 3.4 result in instances (G', H') where both G' and H' are forests, and thus have treewidth of 1. Furthermore, G' has maximum degree of 2, and H' maximum degree of 1. Thus, by Corollary 2.2, we get the following result.

Corollary 4.2. For an instance (G, H) of CF-MM and Cov-MM are para-NP-hard when parameterized by the following parameters: $tw(G)$, $tw(H)$, Δ_G, and Δ_H.

One thing to note here is that any combination of the above mentioned parameters is still a constant, if parameters are. Therefore, any combination of those parameterizations still makes the problems para-NP-hard.

Extended constraint graph

Next we introduce an extended constraint graph for a pair (G, H), denoted as F, which is a graph that on top of the disjunctive constraints also encodes that no two adjacent edges in G can be in a valid matching , where $F := (V_H, E_H \cup E_C)$ with $E_C := \bigcup_{\substack{e_1 \cap e_2 \neq \emptyset \\ e_1 \neq e_2 \\ e_1, e_2 \in E}} \{e_1, e_2\}$. For the conflict and forcing variants the graphs are called the extended conflict and extended forcing graphs respectively.

An aspect to keep in mind is that, in case of forcing constraints, two adjacent edges can be in a forcing constraint together, but are only connected by a single edge in F. In order to utilize the extended forcing graph, one therefore needs to have the initial instance (G, H) by the side to be able to differentiate the two types of constraints.

Having more information to solve the problem encoded in a single graph is quite useful, and the graph can be used to construct an algorithm for both the CF-MM and the Cov-MM problems parameterized by $tw(F)$. The case of CF-MM is the easier one of the two, as the problem amounts to computing the Weighted Maximum Independent Set on F.

Lemma 4.8. Weighted CF-MM admits an algorithm with runtime $\mathcal{O}(2^{tw(F)} \cdot tw(F)^{\mathcal{O}(1)} \cdot n)$.

Proof. Let the pair (G, H) and the weight function $c\colon E_G \mapsto \mathbb{R}_0$ be the input for the CF-MM problem and F its corresponding extended conflict graph with the weight function $c_F\colon V_F \mapsto \mathbb{R}_0$ with $c_F(e) = c(e)$ for $e \in E_G$. This definition ensures that all vertices of F are weighted, because $V_F = V_H = E_G$. It then suffices to use the algorithm for the Weighted Independent Set parameterized by treewidth on F, which would run in the required runtime. One then needs to show that:

There exists a conflict-free matching in (G, H) of weight at least $w \iff$ There exists an independent set of weight at least w in F

($\Rightarrow$) : Let M be a conflict-free matching in (G, H) of weight at least w. Then a set I made of the corresponding vertices to M in F is an independent set, and has total weight of at least w due to the defintion of c_F. No two endpoints of an edge in E_H (conflict constraints) is in I, because M is a conflict-free matching, and such an edge would corresponding to a violated constraint in H. No constraints from E_C (adjacent edge pairs) are violated as well.. By construction, such an edge exists if and only if the corresponding edges in G are adjacent. If both endpoints of some edge of E_C are in I, that would imply that M is not a valid matching.

($\Leftarrow$) : Let I be an independent set of F of weight at least w. Then let M be the set of edges corresponding to the vertices in I. Then M is a conflict-free matching of (G, H) and has weight at least w due to the definition of c_F. Using analogous arguments, no constraints of H are violated, because H is a subgraph of F, and therefore such a violation would correspond to two endpoints of an edge of F being simulateneously in I, a contradiction. M is also a matching, because by the construction of F it is ensured that the vertices representing the adjacent edges of G

are never simultaneously in an independent set, because they are connected by an edge in E_C. □

Before we show the algorithm for Cov-MM, we outline the implications of the previously described reduction. Basically everything that is known about Maximum Indenepdent Set can be transferred onto CF-MM. For example, there exists a $\frac{\Delta+2}{3}$-approximation algorithm given by Halldersson et al. [12], where Δ is the maximum degree of the input graph. It can be used to yield the following result.

Corollary 4.3. There exists a $(\frac{2\Delta_G+\Delta_H}{3})$-approximation algorithm for CF-MM.

Proof. By construction the maximum degree of a vertex in F is the maximum degree of a vertex in H, or Δ_H, plus the number of adjacent to it edges in G. The number of adjacent edges for an edge would be $2(\Delta_G - 1)$, at most $\Delta_G - 1$ for each endpoint of the edge. Thus, $\Delta_H \leq 2\Delta_G + \Delta_H - 2$. So using the approximation algorithm for the independent set yields an $\frac{\Delta_F+2}{3} \leq \frac{2\Delta_G+\Delta_H}{3}$-approximation ratio. □

One can also approximate CF-MM independent of the degree of G. This is a generalization of the 2-approximation algorithm for CF-MM-$(\mathcal{G}, \Delta_1)$ presented by Darmann et al. [1]. In their algorithm, for the input pair (G, H) where H has maximum degree 1, one first computes the maximum weight matching of G and for each conflict in H of the resulting matching removes the cheapest edge. One can generalise without constricting on the maximum degree of H being 1.

Lemma 4.9. Weighted CF-MM admits an $(\Delta_H + 1)$-approximation algorithm.

Proof. Let (G, H) be the input for the CF-MM problem and c the weight function $c\colon E_H \mapsto \mathbb{R}_0$. Let c' be the analogous weight function, only defined on the vertices of H, namely $c'(e) = c(e)$ for $e \in E_G$. Similarly to the approximation algorithm in case $H \in \Delta_1$, one first computes a weighted maximum matching of G, denoted M. Let $c(M) = c'(M)$ denote the sum of the edge weights in M, and $H' := H[M]$, namely the vertex induced subgraph leaving only the vertices of the corresponding edges of M. One can use the greedy algorithm for the Weighted Maximum Independent Set problem on H' as outlined by Sakai et al. [13]. That algorithm would output an independent set of weight at least $\sum_{v \in H'} \frac{c'(v)}{d_{H'}(v)+1}$, which is at least $\sum_{v \in H'} \frac{c'(v)}{\Delta_{H'}+1} = \frac{c'(M)}{\Delta_{H'}+1} \geq \frac{c(M)}{\Delta_H+1}$. Let α denote the weight of the weighted maximum conflict-free matching of (G, H). Since any conflict-free matching is also a valid matching, its weight cannot exceed the maximum weighted matching of the same graph, and thus the final solution weight is at least $\frac{c(M)}{\Delta_H+1} \geq \frac{\alpha}{\Delta_H+1}$, which results in the desired approximation ratio. □

Now we layout how the extended constraint graph can be used for a parameterized algorithm for Cov-MM. We can use a similar approach as for CF-MM. However, because a matching does not enforce adjacent edges to be in the matching, this makes the problem have two types of restrictions. Due to this, we can no longer just use a vertex cover or an independent set algorithm. We, however, combine the best of both worlds to fit our purpose.

Lemma 4.10. Assuming a tree decomposition of F (extended forcing graph) $D := (B, T)$ given, Cov-MM can be solved in $\mathcal{O}(2^{tw(F)} \cdot (tw(F))^2 \cdot |B|)$.

The algorithm is almost identical to the the algorithm for the Vertex Cover Problem parameterized by treewidth presented by Niedermeier [14]. There are however differences in the implementation and the preprocessing. For the rest of the proof we use t to denote $tw(F)$.

Proof. Let (G, H, c) be the input pair for the Weighted Cov-MM problem, and $D := (B, T)$ the tree decomposition of the extended forcing graph F of width t. Important to keep in mind is that vertices of F are edges of G. From this point on, we refer to them as edges to make understanding easier.

The algorithm described is a dynamic programming algorithm via bottom-up traversal of the tree D. As usual, it is divided into 3 steps: initialization, tree traversal, and solution extraction.

For a bag $Y \in B$, let D_Y be the subtree of D rooted at Y, and V_Y be the set of edges which occur in the bags of D_Y, or formally, $V_Y := \bigcup_{X \in D_Y} X$. For each bag $Y \in B$, we define a function w_Y, such that after processing Y, maps subsets S of Y to the optimal weight of a covering matching for the instance made up only of edges in V_Y, which out of all edges in Y only have those in S. Formally, $w_Y[S]$ is the optimal weight of a covering matching M for the instance $(G[V_Y], H[V_Y])$, such that $M \cap Y = S$, if such a matching exists.

First one needs to initialize the function values for each bag $Y \in B$. This can be done by enumerating all possible vertex covers of $H[Y]$, as well as all independent sets in $F[E_F \backslash E_H]$ (which is the subgraph only consisting of the edges in F between adjacent edges of G), which can be done via iterating through all 2^{t+1} subsets of V_Y and checking whether all of the at most $(t+1)^2$ edges are covered. Therefore, enumerating all vertex covers can be done in $\mathcal{O}(2^t \cdot t^2)$. Since for a vertex cover C, its complements in the subgraph of the bag Y, namely $H[Y] - C$, is an independent set, we can also use the same procedure to enumerate all independent sets. Then we create two boolean arrays with entries for each subset indicating whether a particular subset is a vertex cover/independent set. This can be done in a single traversal, and thus takes time $\mathcal{O}(2^t)$ for each bag. With the two arrays, checking whether a subset is a valid covering matching can be done by checking whether the entry for the subset in both arrays is set to True. After having all vertex covers and independent sets, one can iterate through all subsets $M_Y \subseteq Y$ and set the function values according to:

$$w_Y[M_Y] := \begin{cases} c[M_Y] & \text{if } M_Y \text{ is a covering matching of } (G[Y], H[Y]) \\ -\infty & \text{otherwise} \end{cases}$$

Since there are $\mathcal{O}(|B|)$ bags in D, initializing the function would take $\mathcal{O}(2^t \cdot t^2 \cdot |B|)$ for the whole tree.

After initializing the values, one traverses the tree bottom up, and processes each bag Y iteratively by considering one child at a time. For a child X and for each $M_Y \subseteq Y$ one updates the function values $w_Y[M_Y]$ using the following formula:

$$\begin{aligned} w_Y[M_Y] = & w_Y[M_Y] && (4.1)\\ & + \max\{w_X[M_X] \mid M_X \subseteq X,\ M_X \cap X \cap Y = M_Y \cap X \cap Y\} && (4.2)\\ & - c[M_X \cap M_Y] && (4.3) \end{aligned}$$

The intuition behind the formula is similar to calculating the size of the union of two sets (analogously here of $M_Y \cup M_X$). One sums up the weights of the optimal matchings, while subtracting the weights which were added twice due to an intersection. An example of what the formula computes is illustrated in Section 4.3.

The correctness stems from the fact that for a bag Y with a child bag X, it cannot be the case that there is a constraint between two edges e_Y and e_X with $e_Y \in Y \backslash X$ and $e_X \in X \backslash Y$, nor can the two be adjacent. This is the result of the way tree decompositions are defined. Assume that is case, then e_Y and e_X are neighbours in F (by construction). By definition of tree decomposition, there must be a bag V_{XY}, such that $\{e_X, e_Y\} \subseteq V_{X_Y}$. Since e_X is not in Y and all the bags containing e_X are connected, V_{XY} is in the subtree rooted at X. Similarly with e_Y. Because e_Y is not in X and all the bags containing e_Y are connected, V_{XY} cannot be in the subtree rooted at X, a contradiction.

The processing of each child can be done in the following way. One first sorts Y and X by the canonical ordering of $Y \cap X$. The ordering of $Y \backslash X$ in the table of Y and $X \backslash Y$ in the table of X is irrelevant. This can be done in time $\mathcal{O}(2^t \cdot \log 2^t) = \mathcal{O}(2^t \cdot t)$. After sorting one can traverse both tables in parallel and update the values. This is possible because the two tables are sorted by $Y \cap X$, which ensures that one does not need to traverse the two tables back to find the relevant intersection. This can be done in $\mathcal{O}(2^t \cdot t)$, where $\mathcal{O}(2^t)$ is contributed by the number of entries and $\mathcal{O}(t)$ is the time it would take to compare the two table entries and see whether they coincide at $Y \cap X$. Therefore, it takes $\mathcal{O}(2^t \cdot t)$ to process one child.

Because D is a tree, it has $\mathcal{O}(|B|)$ edges, and thus there are at most $\mathcal{O}(|B|)$ children considered during the bottom-up traversal, which results in the processing time of $\mathcal{O}(2^t \cdot t \cdot |B|)$.

For the root R of D and $S \subseteq R$, $w_R[S]$ stores the weight of the optimal covering matching M of $(G[V_R], H[V_R]) = (G, H)$ (without loss of generality G has no isolated vertices) such that $M \cap R = S$, and thus by going through all of the subsets of R and finding the maximum function value one can extract the solution. That would take additional time $\mathcal{O}(2^t)$.

The runtime for any of the steps does not exceed $\mathcal{O}(2^t \cdot t^2 \cdot |B|)$, which yields the desired runtime. □

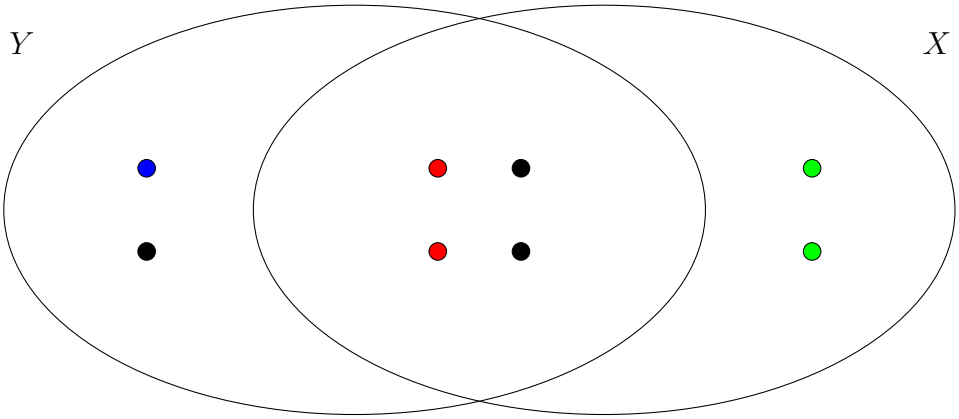


Figure 4.2. An example of the computation in the proof of Lemma 4.10 with a bag Y and its child X. A matching M_Y in Y is made up of blue and red points, a matching M_X in X is made up of green and red points. In order to get the weight of the matching $M_Y \cup M_X$, one adds the weights of M_Y and M_X, and substracts $M_X \cap M_Y$ (red points), as their weights were added twice. This is, however, a simplified version and does not reflect the whole picture. The weights that are used in Lemma 4.10 are not of the matchings M_Y and M_X, but rather of the matchings which are extended from these to the whole subtree.

Chapter 5

Conclusion

We have extended previously known results and have shown that both unweighted CF-MM and unweighted Cov-MM are NP-hard on paths even when each edge is part of at most one constraint. In addition, we have studied the feasibility problem of the covering variant, have shown that it is in P, and used that result to design a 2-approximation algorithm, as well as an $\mathcal{O}^*(2^k)$ algorithm, where k is the solution size. There is no subexponential algorithm under ETH. Further examination is needed to improve the approximation factor, or to show that 2 is the lower bound based on additional assumptions, such as P $\neq$ NP or ETH. A better exponential base for the unweighted Cov-MM parameterized by the solution size is also of interest. In addition, the existence of an FPT for Cov-MM poses a question of the optimum kernel size.

We have also considered the size of the maximum matching of G as a parameter, with the conflict version being W[1]-hard and the forcing version being in FPT. We also proposed an extended constraint graph, which allows for both variants to be in FPT parameterized by its treewidth. However, the maximum matching of the constraint graph as a parameter has not been touched upon and requires further examination.

Bibliography

[1] Andreas Darmann, Ulrich Pferschy, Joachim Schauer, and Gerhard J. Woeginger. "Paths, trees and matchings under disjunctive constraints". In: *Discrete Applied Mathematics* 159.16 (2011). 8th Cologne/Twente Workshop on Graphs and Combinatorial Optimization (CTW 2009), pp. 1726–1735. ISSN: 0166-218X. DOI: https://doi.org/10.1016/j.dam.2010.12.016.

[2] Akanksha Agrawal, Pallavi Jain, Lawqueen Kanesh, and Saket Saurabh. "Parameterized Complexity of Conflict-Free Matchings and Paths". In: *Algorithmica* 82.7 (July 2020), pp. 1939–1965. ISSN: 0178-4617. DOI: 10.1007/s00453-020-00681-y.

[3] Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Daniel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. 1st. Springer Publishing Company, Incorporated, 2015.

[4] Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. *Kernelization: Theory of Parameterized Preprocessing*. Cambridge University Press, 2019.

[5] Juraj Hromkovic. *Algorithmics for Hard Problems*. 2nd ed. Texts in theoretical computer science. Springer-Verlag, 2004.

[6] Jörg Flum and Martin Grohe. "Parameterized Complexity Theory". In: *Texts in Theoretical Computer Science. An EATCS Series*. 2006.

[7] Herbert Fleischner, Gert Sabidussi, and Vladimir I. Sarvanov. "Maximum independent sets in 3- and 4-regular Hamiltonian graphs". In: *Discrete Mathematics* 310.20 (2010). Graph Theory — Dedicated to Carsten Thomassen on his 60th Birthday, pp. 2742–2749. ISSN: 0012-365X. DOI: https://doi.org/10.1016/j.disc.2010.05.028.

[8] Paul Erdös and Paul Joseph Kelly. "The Minimal Regular Graph Containing a Given Graph". In: *American Mathematical Monthly* 70 (1963), p. 1074.

[9] David P. Williamson and David B. Shmoys. *The Design of Approximation Algorithms*. Cambridge: Cambridge University Press, 2011. DOI: 10.1017/CBO9780511921735.

[10] Bengt Aspvall, Michael F. Plass, and Robert Endre Tarjan. "A linear-time algorithm for testing the truth of certain quantified boolean formulas". In: *Information Processing Letters* 8.3 (1979), pp. 121–123. ISSN: 0020-0190. DOI: https://doi.org/10.1016/0020-0190(79)90002-4.

[11] Silvio Micali and Vijay V. Vazirani. “An O(v|v| c |E|) algoithm for finding maximum matching in general graphs”. In: *21st Annual Symposium on Foundations of Computer Science (sfcs 1980)*. 1980, pp. 17–27. DOI: 10.1109/SFCS.1980.12.

[12] Magnús Halldórsson and Jaikumar Radhakrishnan. “Greed is good: approximating independent sets in sparse and bounded-degree graphs”. In: *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing*. STOC ’94. Montreal, Quebec, Canada: Association for Computing Machinery, 1994, pp. 439–448. DOI: 10.1145/195058.195221.

[13] Shuichi Sakai, Mitsunori Togasaki, and Koichi Yamazaki. “A note on greedy algorithms for the maximum weighted independent set problem”. In: *Discrete Appl. Math.* 126.2–3 (Mar. 2003), pp. 313–322. ISSN: 0166-218X. DOI: 10.1016/S0166-218X(02)00205-6.

[14] Rolf Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, Feb. 2006. DOI: 10.1093/acprof:oso/9780198566076.001.0001.

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Eigenständigkeitserklärung

Die unterzeichnete Eigenständigkeitserklärung ist Bestandteil jeder während des Studiums verfassten schriftlichen Arbeit. Eine der folgenden zwei Optionen ist **in Absprache mit der verantwortlichen Betreuungsperson** verbindlich auszuwählen:

☒ Ich erkläre hiermit, dass ich die vorliegende Arbeit eigenverantwortlich verfasst habe, namentlich, dass mir niemand beim Verfassen der Arbeit geholfen hat. Davon ausgenommen sind sprachliche und inhaltliche Korrekturvorschläge der Betreuungsperson. Es wurden keine Technologien der generativen künstlichen Intelligenz[1] verwendet.

☐ Ich erkläre hiermit, dass ich die vorliegende Arbeit eigenverantwortlich verfasst habe. Dabei habe ich nur die erlaubten Hilfsmittel verwendet, darunter sprachliche und inhaltliche Korrekturvorschläge der Betreuungsperson sowie Technologien der generativen künstlichen Intelligenz. Deren Einsatz und Kennzeichnung ist mit der Betreuungsperson abgesprochen.

Titel der Arbeit:

Maximum Matching with Disjunctive Constraints

Verfasst von:
Bei Gruppenarbeiten sind die Namen aller Verfasserinnen und Verfasser erforderlich.

Name(n):	**Vorname(n):**
Latifova	Leyla

Ich bestätige mit meiner Unterschrift:
- Ich habe mich an die Regeln des «Zitierleitfadens» gehalten.
- Ich habe alle Methoden, Daten und Arbeitsabläufe wahrheitsgetreu und vollständig dokumentiert.
- Ich habe alle Personen erwähnt, welche die Arbeit wesentlich unterstützt haben.

Ich nehme zur Kenntnis, dass die Arbeit mit elektronischen Hilfsmitteln auf Eigenständigkeit überprüft werden kann.

Ort, Datum	**Unterschrift(en)**
Zürich, 23.03.2026	

Bei Gruppenarbeiten sind die Namen aller Verfasserinnen und Verfasser erforderlich. Durch die Unterschriften bürgen sie grundsätzlich gemeinsam für den gesamten Inhalt dieser schriftlichen Arbeit.

[1] Für weitere Informationen konsultieren Sie bitte die Webseiten der ETH Zürich, bspw. https://ethz.ch/de/die-eth-zuerich/lehre/ai-in-education.html und https://library.ethz.ch/forschen-und-publizieren/Wissenschaftliches-Schreiben-an-der-ETH-Zuerich.html (Änderungen vorbehalten).