Universidade do Estado do Rio de Janeiro

Centro de Tecnologia e Ciências

Instituto de Matemática e Estatística

Bruno Porto Masquio

Connected and Disconnected Matchings

Rio de Janeiro

2022

Bruno Porto Masquio

Connected and Disconnected Matchings

Tese apresentada como requisito parcial para obtenção do título de Doutor, ao Programa de Pós-Graduação em Ciências Computacionais, da Universidade do Estado do Rio de Janeiro.

Orientadores: Prof. Dr. Paulo Eustáquio Duarte Pinto
Prof. Dr. Jayme Luiz Szwarcfiter

Rio de Janeiro
2022

CATALOGAÇÃO NA FONTE

UERJ/REDE SIRIUS/BIBLIOTECA CTC/A

M413 Masquio, Bruno Porto.
Connected and disconnected matchings/ Bruno Porto Masquio. – 2022.
102 f.: il.

Orientadores: Paulo Eustáquio Duarte Pinto, Jayme Luiz Szwarcfiter
Tese (Doutorado em Ciências Computacionais) - Universidade do Estado do Rio de Janeiro, Instituto de Matemática e Estatística.

1. Grafos - Teoria - Teses. 2. Algoritmos - Teses. I. Pinto, Paulo Eustáquio Duarte. II. Szwarcfiter, Jayme Luiz. III. Universidade do Estado do Rio de Janeiro. Instituto de Matemática e Estatística.IV. Título.

CDU 004.42

Patricia Bello Meijinhos CRB7/5217 - Bibliotecária responsável pela elaboração da ficha catalográfica

Bruno Porto Masquio

Connected and Disconnected Matchings

Tese apresentada como requisito parcial para obtenção do título de Doutor, ao Programa de Pós-Graduação em Ciências Computacionais, da Universidade do Estado do Rio de Janeiro.

Aprovada em 18 de Novembro de 2022
Banca Examinadora:

Prof. Dr. Paulo Eustáquio Duarte Pinto (Orientador)
Instituto de Matemática e Estatística - UERJ

Prof. Dr. Jayme Luiz Szwarcfiter (Orientador)
Instituto de Matemática e Estatística - UERJ

Prof. Dr. Luerbio Faria
Instituto de Matemática e Estatística - UERJ

Dr. Guilherme de Castro Mendes Gomes
Google

Prof.ª Dra. Sulamita Klein
Universidade Federal do Rio de Janeiro - UFRJ

Prof. Dr. Fabiano de Souza Oliveira
Instituto de Matemática e Estatística - UERJ

Prof. Dr. Vinicius Fernandes dos Santos
Universidade Federal de Minas Gerais - UFMG

Rio de Janeiro

2022

RESUMO

MASQUIO, Bruno Porto. *Emparelhamentos Conexos e Desconexos*. 2022. 102 f. Tese (Doutorado em Ciências Computacionais) – Instituto de Matemática e Estatística, Universidade do Estado do Rio de Janeiro, Rio de Janeiro, 2022.

Problemas de emparelhamentos em grafos vêm sendo estudados há muito tempo, tendo importantes resultados tanto teóricos quanto práticos. Ao longo das décadas, variações de problemas de emparelhamentos foram estudadas. Algumas dessas podem ser resolvidas em tempo polinomial, enquanto outras não, a menos que $\mathsf{P} = \mathsf{NP}$. Nesta tese, apresentamos brevemente a história dos emparelhamentos e algumas de suas variações, juntamente com seu estado da arte. Também apresentamos resultados em uma dessas variações: $\mathscr{P}$-emparelhamentos. Um emparelhamento M é um $\mathscr{P}$-emparelhamento se o subgrafo induzido pelas extremidades das arestas de M satsifaz a propriedade $\mathscr{P}$. Como exemplos, para escolhas apropriadas de $\mathscr{P}$, os problemas Emparelhamento Induzido, Emparelhamento Unicamente Restrito, Emparelhamento Acíclico, Emparelhamento Conexo e Emparelhamento Desconexo surgem. Para muitos desses problemas, sabe-se que encontrar um $\mathscr{P}$-emparelhamento de cardinalidade máxima é um problema NP-difícil. Duas exceções, que são foco desta tese, são emparelhamentos conexos e emparelhamentos desconexos, em que a propriedade $\mathscr{P}$ é que o subgrafo induzido pelo emparelhamento seja conexo ou desconexo, respectivamente. Enquanto podemos obter um emparelhamento conexo máximo em tempo polinomial, a complexidade de encontrar um emparelhamento desconexo máximo ainda era desconhecida, apesar de perguntada em 2005 por Goddard, Hedetniemi, Hedetniemi e Laskar [Generalized subgraph-restricted matchings in graphs, Discrete Mathematics, 293 (2005) 129 – 138]. Nesta tese, respondemos essa pergunta. De fato, consideramos o problema de forma generalizada, em que deseja-se encontrar c-emparelhamentos desconexos; para tais emparelhamentos, os subgrafos induzidos pelos seus conjuntos de vértices devem possuir pelo menos c componentes conexas. Mostramos que, para todo $c \geq 2$ fixo, esse problema é NP-completo mesmo se restringirmos a entrada para grafos bipartidos de diâmetro limitado, enquanto que pode ser resolvido em tempo polinomial para $c = 1$. Para o caso em que c é parte da entrada, mostramos que o problema é NP-completo para grafos cordais e grafos com grau limitado, podendo ser resolvido em tempo polinomial para grafos de intervalo. Exploramos, também, a complexidade parametrizada do problema. Apresentamos um algoritmo FPT para o parâmetro treewidth e um algoritmo XP para grafos com um número polinomial de separadores minimais quando parametrizado por c. Complementamos esses resultados mostrando que, a menos que $\mathsf{NP} \subseteq \mathsf{coNP/poly}$, o problema relacionado Emparelhamento Induzido não admite kernel polinomial quando parametrizado pela cobertura de vértices e pelo tamanho do emparelhamento, nem quando parametrizamos pela distância para clique e tamanho do emparelhamento. Para emparelhamentos conexos, apresentamos um algoritmo de tempo linear para encontrar emparelhamentos conexos máximos e um emparelhamento máximo é dado na entrada. Então, voltamos nossa atenção para uma generalização de emparelhamentos conexos que considera grafos ponderados em arestas e cujo problema chamamos de Emparelhamento Conexo Ponderado Máximo. Esse problema foi motivado pelo problema clássico Emparelhamento Ponderado Máximo, estudado há décadas, e

com várias aplicações, incluindo o problema bem conhecido Atribuição. Além disso, alguns outros $\mathscr{P}$-emparelhamentos, tais como acíclicos e induzidos, também já foram considerados para grafos ponderados. No Emparelhamento Conexo Ponderado Máximo, queremos encontrar um emparelhamento M tal que as extremidades das suas arestas induzem um subgrafo conexo e a soma dos pesos das arestas de M é máxima. Ao contrário do problema não ponderado Emparelhamento Conexo, que pode ser resolvido em tempo polinomial para grafos gerais, mostramos que Emparelhamento Conexo Ponderado Máximo é NP-difícil mesmo para grafos bipartidos de diâmetro limitado, grafos starlike, grafos planares bipartidos e grafos subcúbicos planares, enquanto pode ser resolvido em tempo linear para árvores e grafos com grau máximo dois. Quando restringimos as arestas para terem pesos somente não negativos, mostramos que o problema pode ser resolvido em tempo polinomial para grafos cordais, enquanto que continua NP-difícil para a maior parte dos casos. Também estudamos a complexidade parametrizada do Emparelhamento Conexo Ponderado Máximo. Como resultados positivos, apresentamos um algoritmo de tempo exponencial parametrizado pela treewidth. Em relação a kernelização, mostramos que, quando restringimos os pesos a valores binários, o seu problema de decisão, Emparelhamento Conexo Ponderado, não admite kernel polinomial quando parametrizado pelo tamanho da cobertura de vértices sob hipóteses teóricas de complexidade padrão.

Palavras-chave: Algoritmos. Complexidade. Emparelhamento. Grafos bipartidos. Grafos cordais. Emparelhamento Desconexo. Emparelhamento Conexo.

ABSTRACT

MASQUIO, Bruno Porto. *Connected and Disconnected Matchings*. 2022. 102 f. Tese (Doutorado em Ciências Computacionais) – Instituto de Matemática e Estatística, Universidade do Estado do Rio de Janeiro, Rio de Janeiro, 2022.

Matching problems in graphs have been studied for a long time, achieving important results in both theoretical and practical aspects. Over the decades, many variations of matching problems were studied. Some of them can be solved in polynomial time, while others can not, unless P = NP. In this thesis, we briefly present the history of matchings and a survey of some of its variations along with its state of the art. We also give results on one of these variations: $\mathscr{P}$-matchings. A matching M is a $\mathscr{P}$-matching if the subgraph induced by the endpoints of the edges of M satisfies property $\mathscr{P}$. As examples, for appropriate choices of $\mathscr{P}$, the problems Induced Matching, Uniquely Restricted Matching, Acyclic Matching, Connected Matching and Disconnected Matching arise. For many of these problems, finding a maximum $\mathscr{P}$-matching is a knowingly NP-hard problem. Two exceptions, that are the focus of this thesis, are connected matchings and disconnected matchings, where $\mathscr{P}$ is that the graph is connected and disconnected, respectively. While we can obtain a maximum connected matching in polynomial time, the complexity of finding a maximum disconnected matching was still unknown, though asked since 2005 by Goddard, Hedetniemi, Hedetniemi, and Laskar [Generalized subgraph-restricted matchings in graphs, Discrete Mathematics, 293 (2005) 129 – 138]. In this thesis, we answer this question. In fact, we consider the generalized problem of finding c-disconnected matchings; such matchings are ones whose vertex sets induce subgraphs with at least c connected components. We show that, for every fixed $c \geq 2$, this problem is NP-complete even if we restrict the input to bounded diameter bipartite graphs, while can be solved in polynomial time if $c = 1$. For the case when c is part of the input, we show that the problem is NP-complete for chordal graphs and bounded degree graphs while being solvable in polynomial time for interval graphs. Then, we explore the parameterized complexity of the problem. We present an FPT algorithm under the treewidth parameterization, and an XP algorithm for graphs with a polynomial number of minimal separators when parameterized by c. We complement these results by showing that, unless NP $\subseteq$ coNP/poly, the related Induced Matching problem does not admit a polynomial kernel when parameterized by vertex cover and size of the matching nor when parameterized by vertex deletion distance to clique and size of the matching. As for connected matchings, we give a linear-time algorithm for finding a maximum connected matching if a maximum matching is given. So, we turn our attention to a generalization of connected matchings that considers edge-weighted graphs and whose problem we name Maximum Weight Connected Matching. This problem is motivated by the usual Maximum Weight Matching, studied for decades, and with many applications, including the well-known Assignment problem. Besides, some other $\mathscr{P}$-matchings, such as acyclic matchings and induced matchings, were also considered under weighted graphs before. In Maximum Weight Connected Matching, we want to find a matching M such that the endpoints of its edges induce a connected subgraph and the sum of the edge weights of M is maximum. Unlike the unweighted Connected Matching problem, which can be solved in polynomial time for

general graphs, we show that Maximum Weight Connected Matching is NP-hard even for bounded diameter bipartite graphs, starlike graphs, planar bipartite graphs, and subcubic planar graphs, while solvable in linear time for trees and graphs having a degree at most two. When we restrict edge weights to be non-negative only, we show that the problem turns out to be polynomially solvable for chordal graphs, while it remains NP-hard for most of the other cases. We also consider the parameterized complexity of Maximum Weight Connected Matching. On the positive side, we present a single exponential time algorithm when parameterized by treewidth. As for kernelization, we show that, even when restricted to binary weights, its decision problem, Weighted Connected Matching, does not admit a polynomial kernel when parameterized by vertex cover number under standard complexity-theoretical hypotheses.

Keywords: Algorithms. Complexity. Matching. Bipartite graphs. Chordal graphs. Disconnected Matching. Connected Matching.

LIST OF FIGURES

LIST OF TABLES

CONTENTS

INTRODUCTION

A graph is a mathematical tool that represents elements, by vertices, and connections between them, by edges. Although the concept may seem too general, it helps to find solutions to many real-world problems in different fields, like physics, biology, geography, and on information systems. For this reason, much research has been done on this subject of computer science.

Along with the study of graphs, some concepts around them arise, including matchings; it consists in disjointly pairing vertices that are connected by edges. Often, we are interested in finding the largest possible matching in a graph, which may be applied to some practical problems.

For instance, suppose we have a group of workers and other of tasks, in such a way that a worker is able to do only some tasks, which are known previously. To optimize resources, an employer would assign the maximum number of workers to perform one task each, in order to minimize the number of idle workers. For this purpose, we could randomly make assignments until there are no more possibilities; with luck, or with lots of tries, we would make the right choices and assign the maximum number of workers, but it is not guaranteed or may take too long. So, one could ask if there is a fast method that always finds an optimal assignment in this situation.

The answer is yes, and it can be done by graph modeling. As such, let each worker and each task be a vertex, with edges connecting each worker to the tasks he is able to perform. In this case, we would seek a matching with the highest cardinality in this graph, whose problem is also known as Maximum Matching. Finally, every matched edge would represent an assignment, which is optimal.

This method, which we will call from now on an algorithm, even though may be guaranteed to find a solution to our problem, also has to end within a convenient period of time. For instance, it is not satisfactory if we design an algorithm that takes days to solve Maximum Matching for a small group of workers and tasks. Moreover, this algorithm also has to perform relatively well as these groups get bigger. If it does, we say that the algorithm is efficient.

At this point, the interest in studying computational complexity theory arises, which will categorize problems in classes depending on some standard complexity characteristics. If it is known that a problem can be solved by an efficient algorithm, we say that it is solvable in polynomial time. Frequently, we can also prove that a problem is NP-hard, that is, most likely, it does not admit an efficient algorithm. Of course, this is an oversimplification of all the vast theory behind this fact, which can be found in different books, including one by Michael R. Garey and David S. Johnson called Computers and Intractability: A Guide to the Theory of NP-Completeness [78].

When dealing with some problems, sometimes it is convenient to consider only graphs with a certain structure, which are known as graph classes. Restricting our problem this way often helps to find efficient algorithms to solve problems. In our example with workers and tasks, note that the graph does not have edges between two workers or two

tasks. This is actually a very common graph class, named bipartite graphs. Although Maximum Matching itself does not seem to get easier in bipartite graphs compared to general ones, other problems do. One example is Vertex Cover, which is NP-hard on general graphs, but solvable in polynomial time for bipartite graphs.

Now, let us focus on studies of matchings, which have begun a long time ago. Although it is hard to be precise about the first time it appeared, one remarkable contribution was made in 1891, when Petersen's Theorem was presented. This was also one of the first results in graph theory, and it characterized some graphs that admit matchings that could cover all vertices. These kinds of matchings are also known as perfect matchings, which prompted many other studies over the decades.

The lots of characterizations and applications that were given around matchings later motivated some scientists to expand the concept of it by considering some constraints or generalizations. This thesis is focused on a restricted type of matching, known as $\mathscr{P}$-matching; in this matching, the subgraph induced by the endpoints of the matched edges has to satisfy property $\mathscr{P}$.

Among the first to consider $\mathscr{P}$-matchings were Stockmeyer and Vazirani [168] in 1982, as they studied matchings in which edges have a certain distance between them. For a specific distance, the corresponding matching was later known as induced matching or strong matching. In particular, an induced matching requires that the corresponding induced subgraph must have no edges between vertices, except the ones contained in the matching.

Later, other kinds of matchings were later defined and studied, such as uniquely restricted matchings [82], acyclic matchings [81], disconnected matchings [81], and connected matchings [81]. These last two, proposed by Goddard et al. [81], consider the properties of connectivity of the corresponding subgraph. More specifically, connected and disconnected matchings require, respectively, that the induced subgraph is connected or disconnected. In the same article, the authors left the question concerning the complexity of the Disconnected Matching problem.

Motivated by this, in this thesis we answer this question and study the complexity of Disconnected Matching and Connected Matching. We also go further about the latter, approaching its weighted variant. We proved the complexity of the problems studied, which was not known before. Besides, we found some interesting contrasts with their complexity when we apply some constraints to the problems, including results on graph classes.

Our contributions and structure of this thesis

This thesis is organized as follows:

- The remainder of this chapter includes details of the publications on the work in this thesis and a preliminaries section, which defines most of the notation and concepts used in this thesis.

- In Chapter 1, we briefly consider the history of matchings, presenting some of the main results and definitions obtained on this subject. This includes Petersen's Theorem, König-Egerváry Theorem, Hall's Marriage Theorem, Tutte's Theorem, Gallai-Edmonds Structure, Berge's Theorem. Then, we make a survey including some results and the state of the art of various matching variations. Among them, we mention weighted matchings, minimum maximal matchings, rank-maximal

matchings, and stable marriage problem. We also survey the $\mathscr{P}$-matching and its subtopics: induced matchings, uniquely restricted matchings, acyclic matchings, connected matchings, and disconnected matchings.

- In Chapter 2, we describe the problem and discuss about disconnected matchings. We show that, for every fixed $c \geq 2$, the corresponding problem is NP-complete for diameter three bipartite graphs. When c is part of the input, we show that the problem is NP-complete for chordal graphs and bounded degree graphs, while being solvable in polynomial time for interval graphs. Then, we present an FPT algorithm under the treewidth parameterization, and an XP algorithm for graphs with a polynomial number of minimal separators when parameterized by c. We prove that the related Induced Matching problem, and consequently Disconnected Matching, do not admit a polynomial kernel when parameterized by vertex cover and size of the matching nor when parameterized by vertex deletion distance to clique and size of the matching. Finally, in Section 2.5, we show hardness results on the problem of deciding if, given a graph, the equality between the i-disconnected matching number and the j-disconnected matching number holds, for $1 \leq i < j$.

- In Chapter 3, we describe the problem and discuss about connected matchings. In Section 3.1, we present a linear algorithm that, given a graph and one of its maximum matchings, outputs a maximum connected matching, as well as an example of its execution. In Section 3.2, we consider weighted connected matchings, showing that its corresponding problem is NP-complete for starlike graphs and subcubic planar graphs. When we restrict the weights to be non-negative, we show that the problem is in P for chordal graphs, while NP-complete for planar bipartite graphs and diameter four bipartite graphs. Then, we give linear-time algorithms for graphs with a maximum degree at most two and for trees. On the other hand, we present a single exponential time algorithm when parameterized by treewidth, and show that it does not admit a polynomial kernel when parameterized by vertex cover number.

- Finally, we present our conclusions of the approached problems, along with some suggestions for future work.

Publications

Our Disconnected Matching results, in Chapter 2 except Section 2.5, as well as our linear algorithm for Maximum Connected Matching, in Section 3.1 were presented at 27th International Computing and Combinatorics Conference (COCOON 2021); it is currently under the review process for publication. Besides, our results for Weighted Connected Matching, from Section 3.2, were presented at 15th Latin American Theoretical Informatics Symposium (LATIN 2022). Soon, we will submit the full paper to a journal. We also presented abstracts and partial results about these subjects at other conferences. We summarize our publications in the following.

- Encontro de Teoria da Computação, 2020, Brasil. Anais do Encontro de Teoria da Computação (ETC 2020) [132].

- LII Simpósio Brasileiro de Pesquisa Operacional (SBPO 2020) [134].

- 9th Latin American Workshop on Cliques in Graphs (LAWCG 2020) [135].

- Congresso Nacional de Matemática Aplicada e Computacional (CNMAC 2021) [136].
- Encontro de Teoria da Computação, 2021, Brasil. Anais do VI Encontro de Teoria da Computação (ETC 2021) [84].
- 27th International Computing and Combinatorics Conference (COCOON 2021) [88].
- Encontro de Teoria da Computação, 2022, Brasil. Anais do VII Encontro de Teoria da Computação (ETC 2022) [85].
- 10th Latin American Workshop on Cliques in Graphs (LAWCG 2022) [86].
- 15th Latin American Theoretical Informatics Symposium (LATIN 2022) [89].
- XXI Latin-Iberoamerican Conference on Operations Research (CLAIO 2022) [87].

All the results presented in this thesis include contributions from authors Guilherme C. M. Gomes, Vinicius F. dos Santos, as well as the advisors Paulo E. D. Pinto and Jayme L. Szwarcfiter.

The complexity result for $\beta_{d,i} = \beta_{d,j}$, in Section 2.5, has not been submitted for publication yet. This work will also include other results, like deciding if the equality $\beta_d = \beta_*$ holds for a given graph. The authors of this paper are Guilherme C. M. Gomes, Bruno P. Masquio, Paulo E. D. Pinto, Dieter Rautenbach, Vinicius F. dos Santos, Jayme L. Szwarcfiter, and Florian Werner.

Preliminaries

Let us present some of the notations and concepts we adopt throughout this thesis. As references, we use standard nomenclatures and basic concepts of graph theory as in [22, 24], complexity theory as in [78], and parameterized complexity as in [38].

For a set C, we say that A,$B \subseteq C$ *partition* C if $A \cap B = \emptyset$ and $A \cup B = C$; we denote a partition of C into A and B by $A \dot{\cup} B = C$. For an integer k, we define $[k] = \{1, \ldots, k\}$.

Graph definitions. In this thesis, we only use finite, simple, and undirected graphs. Let $G = (V,E)$ be a graph and $W \subseteq V$ a subset of its vertices. Sometimes, we also use $V(G)$ and $E(G)$ to denote the set of vertices and edges of G. Moreover, when there is no ambiguity, we use $n = |V(G)|$ and $m = |E(G)|$ for a graph G. An *edge* e is a pair of vertices $\{u,v\}$, which are called *endpoints* of e. We can equivalently write this edge as uv or as vu.

We say that $G[W]$ is the subgraph of G *induced* by $W \subseteq V$. That is, $G[W] = (W, E_W)$, such that E_W contains an edge $e \in E$ if and only if $|e \cap W| = 2$. Also, the operations $G - uv$ and $G - v$ produce, respectively, the graphs $G' = (V,E \setminus \{uv\})$ and $G[V \setminus \{v\}]$. The *degree* of vertex $v \in V$ is the number of edges of E incident to v, and $\Delta(G)$ is the maximum vertex degree among all vertices of G.

Two graphs G and H are *isomorphic* if there is a bijection $f : V(G) \to V(H)$ such that $uv \in E$ if and only if $f(u)f(v) \in E(H)$. In G, a sequence of vertices $v_1 \ldots v_k$ is a *path* if $v_j v_{j+1} \in E(G)$, for every $1 \leq j \leq k-1$. A *cycle* is a path where $k \geq 4$ and $v_k = v_1$. The *length* of a cycle or a path is defined as the number of edges it contains. A graph is *acyclic* if there is no induced subgraph isomorphic to a cycle. The *distance* between two vertices v,w is the length of the shortest path between v and w. The *diameter* of a connected graph G is the longest distance between any pair of vertices u,$v \in V$.

Furthermore, G is *connected* if there is a path between every pair of its vertices, and *disconnected*, otherwise. A *connected component* of G is a subgraph $G[W]$, for a maximal set $W \subseteq V$ subject to $G[W]$ is connected.

The set W is a *separator* of G if $G - W$ has more connected components than G. Moreover, for v,$w \in V$, if v and w are in the same connected component of G but in different connected components of $G - W$, then W is a *$(v - w)$-separator*. A $(v - w)$-separator W is *minimal* if there is no other $(v - w)$-separator $S \subset W$ in G. If v is the only element of a separator W, then v is an *articulation* of G. Also, an edge $e \in E$ is a *bridge* if $G - e$ has more connected components than G.

The *open neighborhood* and *closed neighborhood* of a vertex $u \in V$ are denoted by $N(u)$ and $N[u]$ respectively, where $N(u) = \{w \mid wu \in E\}$ and $N[u] = N(u) \cup \{u\}$. Analogously, we define $N(W) = (\bigcup_{u \in W} N(u)) \setminus W$ and $N[W] = N(W) \cup W$.

A *matching* M is a subset of E where no two of its edges share an endpoint. The set $V(M)$ contains all endpoints of edges of M; these vertices are also called *M-saturated* vertices, or just *saturated* when there is no ambiguity. We say that M is *maximum* if there is no other matching M' of G such that $|M'| > |M|$, and M is *perfect* if $V(M) = V$. In an abuse of notation, we define $G[M] = G[V(M)]$.

A graph $G' = (V',E')$ is an *edge-weighted graph* if there is a function $w : E' \to \mathbb{Z}$ that comes attached to G. In this case, a matching M' of G' is a *maximum weight matching* if there is no other matching of G' having a greater sum of its edge weights. We denote by $w(uv)$ the weight of the edge $uv \in E'$ and define $w(M') = \sum_{uv \in M'} w(uv)$.

Graph classes. A graph G is *complete* if E contains an edge for every pair of vertices of V. In this case, we can write this graph as K_n, $n = |V|$. A subset $S \subseteq V(G)$ is a *clique* if $G[S]$ is complete. If G is acyclic, then it is called a *forest*; if it is acyclic and connected, it is a *tree*.

The set W is an *independent set* if $G[W]$ has no edges. A graph G is *bipartite* if its vertices can be partitioned into two independent sets V_1 and V_2. In this case, we also use the notation $G = (V_1 \dot\cup V_2,E)$. When E contains all possible edges between elements of V_1 and V_2, we say that G is a *complete bipartite graph*. We denote by $K_{a,b}$ the complete bipartite graph with a vertices in one part and b vertices in the other. A *star* is a graph isomorphic to $K_{1,b}$, for some b.

If all vertices of G have degree k, then G is *k-regular*. For the case where $k = 3$, we say that G is *cubic*. On the other hand, if $\Delta(G) \leq 3$, then G is *subcubic*.

If there is no induced subgraph of G that is isomorphic to H, then G is *H-free*. We denote by P_n and C_n the *path* and *cycle graphs*, that are respectively isomorphic to a path and a cycle with n vertices.

A graph G is *chordal* if it has no induced cycle with more than three vertices. Also, if G is chordal, then every minimal separator is complete [44]. We can represent a chordal graph with a tree, called a *clique tree* [172] [123]. A clique tree of G is a tree T representing G in which vertices and edges of T correspond, respectively, to maximal cliques and minimal separators of G. There exist linear time algorithms that generate a clique tree from a given chordal graph [18] [25] [165] [14]. A graph is an *interval graph* if it is the intersection graph of intervals on a line; equivalently, it is an interval graph if it is chordal and admits a clique tree isomorphic to a path. A graph is a *starlike* graph if it is chordal and has a clique tree that is a star graph. A graph is *planar* if it can be embedded on the plane without edge crossings.

Parameterized complexity. A parameterized problem Γ is said to be XP when

parameterized by k if it admits an algorithm running in $f(k)n^{g(k)}$ time for computable functions f,g; it is said to be FPT when parameterized by k if $g \in \mathcal{O}(1)$.

A *kernelization algorithm*, or simply a *kernel*, for a parameterized problem Q is an algorithm $\mathcal{A}$ that, given an instance (I,k) of Q, works in polynomial time and returns an equivalent instance (I',k') of Q. Moreover, we require that size $\mathcal{A}(k) \leq g(k)$ for some computable function $g : \mathbb{N} \to \mathbb{N}$.

We say that an NP-hard problem Q *OR-cross-composes* into a parameterized problem Γ if, given t instances $\{P_1, \ldots, P_t\}$ of Π, we can build, in time polynomial on $N = \sum_{i \in [t]} |P_i|$, an instance (x, k) of Γ such that: (i) $k \leq \mathsf{poly}(\max\{|P_i| \mid i \in [t]\} \log t)$ and (ii) (x, k) admits a solution if and only if at least one P_i admits a solution.

In the following, we present *treewidth* concept, which is often considered as a parameter on parameterized algorithms. A *tree decomposition* of a graph G is a pair $\mathbb{T} = (T, \mathcal{B} = \{B_j \mid j \in V(T)\})$, where T is a tree and $\mathcal{B} \subseteq 2^{V(G)}$ is a family where: $\bigcup_{B_j \in \mathcal{B}} B_j = V(G)$; for every edge $uv \in E(G)$ there is some B_j such that $\{u,v\} \subseteq B_j$; for every $i,j,q \in V(T)$, if q is in the path between i and j in T, then $B_i \cap B_j \subseteq B_q$. Each $B_j \in \mathcal{B}$ is called a *bag* of the tree decomposition. G has treewidth at most t if it admits a tree decomposition such that no bag has more than $t + 1$ vertices. For further properties of treewidth, we refer to [159]. We say that $\mathbb{T}$ of G is *nice* if its tree is rooted at, say, the empty bag $r(T)$ and each of its bags is from one of the following four types:

1. *Leaf node*: a leaf x of $\mathbb{T}$ with $B_x = \emptyset$.
2. *Introduce vertex node*: an inner bag x of $\mathbb{T}$ with one child y such that $B_x \setminus B_y = \{u\}$.
3. *Forget node*: an inner bag x of $\mathbb{T}$ with one child y such that $B_y \setminus B_x = \{u\}$.
4. *Join node*: an inner bag x of $\mathbb{T}$ with two children y,z such that $B_x = B_y = B_z$.

For other definitions about parameterized complexity and algorithms, we refer to the book Parameterized Algorithms [38].

1 MATCHINGS AND ITS VARIATIONS

In this chapter, we describe matching concepts and their related problems that arose over the decades.

Let $G = (V,E)$ be a simple graph, $n = |V|$, and $m = |E|$. A *matching* M is a subset of edges of E such that no two edges of M share a common endpoint. Edges contained in M are called *matched edges*. A vertex $v \in V$ is called *M-saturated*, or just *saturated* if v is an endpoint of some edge of M, and, otherwise, *non-saturated*. Note that an empty set is a matching, as well as any subset $M' \subseteq M$.

A matching M is said to be *maximal* if any edge that is not matched has an endpoint in M. That is, there is no matching M' such that $M' \supset M$. Nevertheless, among all matchings of a graph G, if M is a matching with the largest possible cardinality, we say that M is *maximum*. Finally, we say that M is *perfect* if every vertex of the graph is M-saturated. Observe that a graph may not admit a perfect matching, but if it does, every maximum matching is a perfect matching and vice versa.

Graphically, we represent saturated vertices as black-filled circles and matched edges as bold lines. See the maximal matching that is not maximum $\{bc, de\}$ in Figure 1a. On the other hand, the matching $\{ac, be, df\}$ in Figure 1b is perfect and maximum.

Important concepts that come from matching theory are *alternating paths* and *augmenting paths*; many algorithms for maximum matchings are based on it. If M is a matching, a path P is said to be *M-alternating*, or just alternating, if edges of P are alternately in $E \setminus M$ and in M. If the first and last vertices of an M-alternating path P are not saturated, then P is also an *M-augmenting path*, or just augmenting path. In Figure 1a, note that *abc* and *acbedf* are alternating paths, but only the latter is also an augmenting path.

One of the first mentions of a matching problem and also one of the first results in graph theory was in 1891, by Julius Petersen, known as Petersen's theorem.

Theorem 1 ([152])**.** Cubic graphs with at most two bridges have a perfect matching.

Note that two is in fact an upper bound for the number of allowed bridges in the graph. Figure 2a illustrates a cubic graph with three bridges that does not admit a perfect matching.

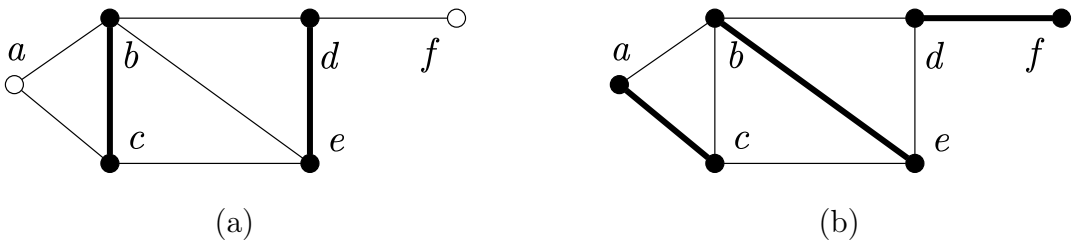
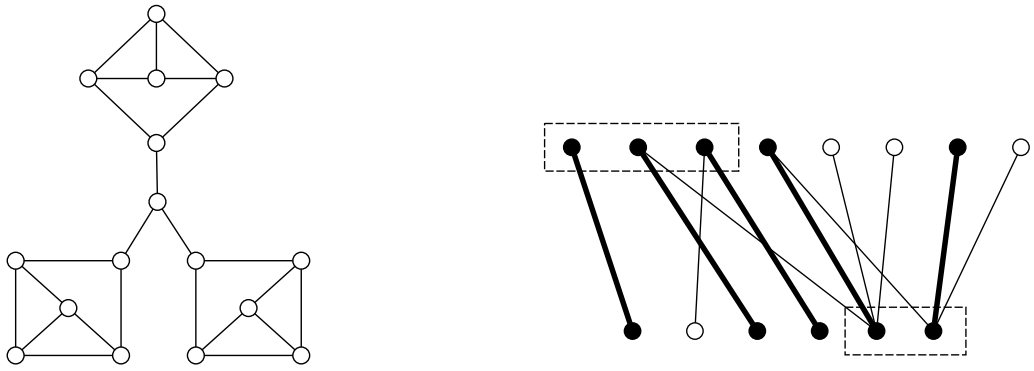


Figure 1 – A maximal matching (a) and a perfect matching (b).



(a) (b)

Figure 2 – A cubic graph with three bridges that does not admit perfect matchings (a). In (b), a maximum matching and a minimum vertex cover of the same size.

Research in matchings became increasingly more common, due to its importance. Another interesting result is the close relationship between a maximum matching and a minimum *vertex cover* in bipartite graphs. A vertex cover $S \subseteq V$ of a graph $G = (V,E)$ is a set such that every edge of E has an endpoint in at least one element of S. It was observed that these two sets have the same size by the following theorem, which is known as König-Egerváry Theorem or König's Theorem.

Theorem 2 ([53, 120])**.** In a bipartite graph, the size of a maximum matching equals the size of a minimum vertex cover.

This equivalence observed by König and Egerváry can be used to find a minimum vertex cover given a maximum matching in a bipartite graph $G = (V_1 \dot\cup V_2, E)$. For this purpose, let M be a maximum matching of G. Choose one bipartition, say V_1. Now, let S be the set of all vertices that are contained in alternating paths starting from $V_1 \setminus V(M)$. A minimum vertex cover of G can be defined as $(V_2 \cap S) \cup (V_1 \setminus S)$. See Figure 2b as an example of a maximum matching and a minimum vertex cover of the illustrated graph.

Since Maximum Matching can be solved in polynomial time for bipartite graphs (see Section 1.1), the same is also true for Vertex Cover [78]. However, for general graphs, Maximum Matching is in P, and Vertex Cover is NP-complete, even for planar graphs of maximum degree 3 [77].

There has also been interest in characterizing bipartite graphs that admit a matching that saturates all vertices of one part of the bipartition, denoted by *complete matching*, or both of the parts, i.e. perfect matching. In this context, Philip Hall stated that those matchings depend on the neighborhood of every possible subset of bipartitions. His characterization is known as Hall's Marriage Theorem.

Theorem 3 ([94])**.** A bipartite graph $G = (V_1 \dot\cup V_2, E)$ admits a matching that saturates all vertices of V_1 if and only if $|N(X)| \geq |X|$, for every $X \subseteq V_1$.

This theorem can also be used to characterize a bipartite graph that admits perfect matchings. In this case, we can apply Theorem 3 to both of its parts as V_1. This idea immediately leads to the next corollary.

Corollary 1 ([94])**.** A bipartite graph $G = (V_1 \dot\cup V_2, E)$ admits a perfect matching if and only if $|N(X)| \geq |X|$, for every $X \subseteq V_1$ and $X \subseteq V_2$.

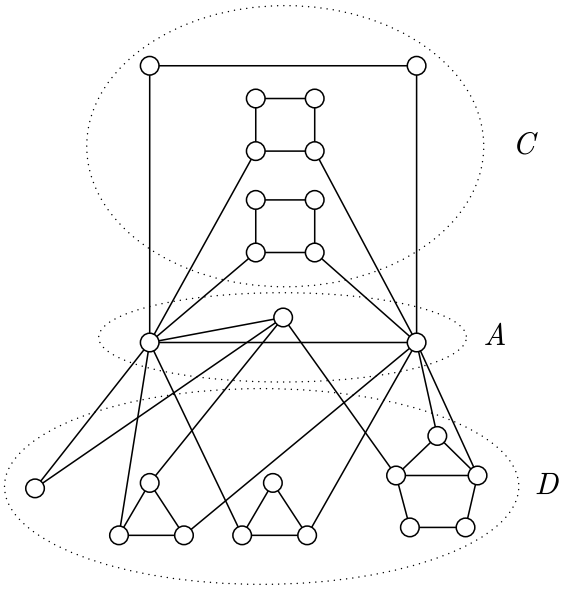


Figure 3 – Example of a Gallai-Edmonds decomposition.

If the graph is not bipartite, we cannot make use of Hall's Marriage Theorem. So, one should find another characterization for general graphs. This issue was only solved by William Thomas Tutte in 1947. This result has been given the name of Tutte's Theorem [173]. It is also presented in the book Graph Theory with Applications [21], whose proof given is based on [128].

Theorem 4 ([173])**.** A graph $G = (V, E)$ has a perfect matching if and only if for every $U \subseteq V$, the subgraph $G-U$ has at most $|U|$ connected components having an odd number of vertices.

The next theorem gives some characterizations for graphs in terms of some matching properties, based on a partition of its vertices. Given a graph $G = (V,E)$, let $D \subseteq V$ be the set of vertices $v \in V$ where there exists a maximum matching M that does not saturate v. Also, let $A = N(D)$ and $C = V \setminus (A \cup D)$. Given a graph, we can obtain its decomposition in polynomial time using the blossom algorithm [51]. In Figure 3, we present a graph with its corresponding decomposition.

It was proved independently by Tibor Gallai and Jack Edmonds and is known as Gallai-Edmonds Structure.

Theorem 5 ([51, 74, 75])**.** Let G be a graph and D,A,C its corresponding Gallai-Edmonds decomposition. The following properties hold:

- Every odd component H of $G - A$ is hypomatchable[1] and has $V(H) \subseteq D$.
- Every even component H of $G - A$ has a perfect matching and has $V(H) \subseteq C$.
- For every $X \subseteq A$, the set $N(X)$ contains vertices in more than $|X|$ odd components of $G - A$.

[1] A graph G is hypomatchable if $G - v$ has a perfect matching for every $v \in V(G)$.

The next theorem, known as Berge's Theorem, characterizes maximum matchings in terms of the existence of augmenting paths. It is used as a basis of many maximum matching algorithms, discussed in more details in Section 1.1.

Theorem 6 ([13])**.** Let G be a graph. A matching M of G is maximum if and only if G has no augmenting path.

Note that if we find an augmenting path P, we can increase the size of the matching M by the operation $M \bigoplus P = (M \setminus P) \cup (P \setminus M)$. That is, we obtain a matching larger than M by removing the matched edges of P from M and adding to M all remaining edges of P.

For a more detailed history of matching results, we refer to the book Matching Theory [154]. Over the years, variations and problems around matchings contexts we studied. In the next sections of this chapter, we list some of them, giving a short brief history and presenting some related results. Clearly, since the literature is vast, we are not able to cover all related problems.

1.1 Maximum cardinality matchings

Possibly the most important problem involving matching in graphs is finding a maximum matching, which is usually denoted by Maximum Matching. The first polynomial-time algorithms for bipartite graphs were designed in [120, 121], with complexity $\mathcal{O}(mn)$.

The problem for general graphs was only solved in 1965 by Jack Edmonds, who designed a beautiful algorithm known as blossom algorithm, which runs in $\mathcal{O}(mn^2)$ [51]. When dealing with general graphs, some techniques do not work as in bipartite graphs, since they can have odd-length cycles. Given that, Edmond's algorithm works by recursively contracting some odd-length cycles, called blossoms, into a single vertices and finding augmenting paths on the resulting graphs.

Over the years, many other algorithms for general graphs were developed, having increasingly better complexities [58, 65, 66, 110, 122, 180]. The best-known time complexity is $\mathcal{O}(m\sqrt{n})$ [68, 71, 139].

It is worth mentioning that the problem was solved in $\mathcal{O}(m\sqrt{n})$ by an algorithm that works only for bipartite graphs [96]. Although there already exists algorithms with the same complexity for general graphs, it is still important because it is simpler and, hence, more convenient to implement.

The research we referenced showed that we can find a maximum matching very quickly in general graphs. Nevertheless, we can achieve better complexities when restricting the input graph to some classes. Mertzios et al. conducted an excellent survey of results considering Maximum Matching in graph classes [137]. These results are presented in Table 1, where we add the linear-time complexity for block graphs obtained in [133].

Graph Class	Complexity
General	$\mathcal{O}(m\sqrt{n})$ [138]
Interval	$\mathcal{O}(n \log n)^*$ [6] [124]
Circular arcs	$\mathcal{O}(n \log n)$ [124]
Co-interval	$\mathcal{O}(n \log n + m)$ [76]
Convex bipartite	$\mathcal{O}(n)$ [167]
Strongly chordal	$\mathcal{O}(n + m)^\dagger$ [40]
Chordal bipartite	$\mathcal{O}(n + m)$ [34]
Regular	$\mathcal{O}(n^2 \log n)$ [186]
Cographs	$\mathcal{O}(n)^\ddagger$ [184]
Co-comparability	$\mathcal{O}(n + m)$ [137]
Block	$\mathcal{O}(n + m)$ [133]

Table 1 – Complexities of Maximum Matching in some graph classes. For entries marked with $*$, †, ‡ it is required, respectively, an interval representation, a strong perfect elimination order, and a co-tree to be part of the input.

1.2 Weighted Matchings

Another variant, known as Maximum Weight Matching (MWM), considers an edge-weighted graph $G = (V,E)$ and its weight function $\omega : E \to \mathbb{R}$. We want to find a matching M such that the sum of the weights of the edges of M is maximum, that is, a maximum weight matching. When the input graph G is bipartite, the problem is also known as Assignment.

Egerváry generalized Theorem 2, considering edge-weighted graphs [53]; this result was later used to solve the Assignment problem in polynomial time for the first time, with time complexity $\mathcal{O}(n^4)$ [121]. Some algorithms with better complexities for bipartite graphs were later developed. The best complexity obtained so far is $\mathcal{O}(n(n \log n + m))$ [60].

Only years later from the first algorithm for bipartite graphs, a polynomial-time algorithm was designed for general graphs, by Jack Edmonds in 1965 [50]. This algorithm is a generalization of Kuhn's Hungarian algorithm, for weighted matching on bipartite graphs [121].

Over time, several implementations of Edmond's algorithm have been given with increasingly faster running times [8, 64, 65, 73, 122], recently reaching the complexity $\mathcal{O}(n(n \log n + m))$ [69, 70], the same obtained before for bipartite graphs.

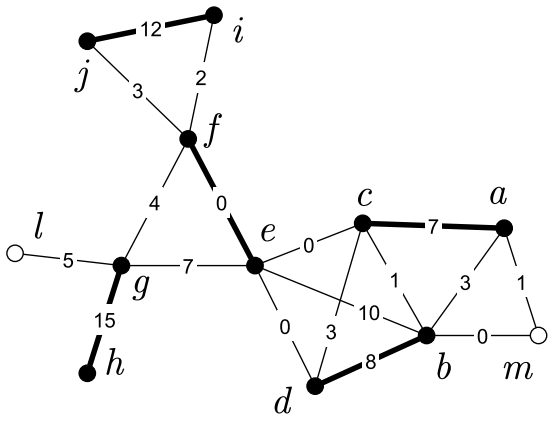


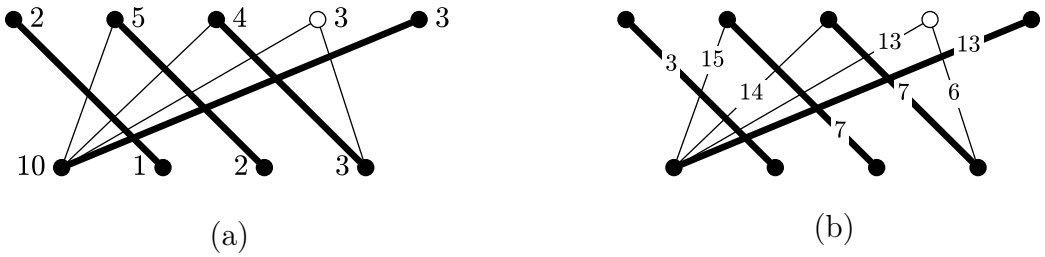
Figure 4 – Example of a maximum weight matching.

1.3 Vertex-Weighted Matchings

Another variant comes when we consider a vertex-weighted graph $G = (V,E)$ and a non-negative weight function $\omega : V \to \mathbb{R}_{\geq 0}$. We want to find a matching M such that the sum of the weights of M-saturated vertices is maximum. This problem is known as Maximum Vertex-Weighted Matching (MVM).

An instance of MVM can be transformed into one of Maximum Weight Matching by assigning each edge a weight obtained by summing the weights of its endpoints. Hence, algorithms for MWM can be used to solve MVM problems [4]. This transformation, though, is not as efficient as running directly an algorithm for MVM.

Note an example of a vertex-weighted graph as well as its maximum weight vertex matching in Figure 5a. On its side, Figure 5b shows the corresponding edge-weighted graph.



(a) (b)

Figure 5 – A vertex-weighted graph with its maximum weight vertex matching (a) and its edge-weighted corresponding instance (b).

Among applications of MVM, we have the design of network switches [170], schedules for training of astronauts [12], computation of sparse bases for the null space or the column space of a rectangular matrix [37, 153, 155], or internet advertising [4].

One of the first mentions of this problem was in 1984, in which the authors presented a polynomial algorithm running in $\mathcal{O}(m\sqrt{n}\log n)$ [164]. Even though this running time is still the best known, this algorithm has been reported as hard to implement. Motivated by this fact, Tabatabaee et al. designed a simpler one that runs in $\mathcal{O}(mn)$ [170].

Moreover, MVM has also been studied from the approximation approach. In 2009, Mahantesh Halappanavar dedicated his Ph.D. thesis to study algorithms for maximum

vertex-weighted matchings [92]. He designed a 2/3-approximation algorithm for bipartite graphs and a 1/2-approximation algorithm for graphs in general. Later, A. Al-Herz and A. Pothen proposed a 2/3-approximation algorithm for general graphs that runs in $\mathcal{O}(m \log \Delta + n \log n)$ [4].

1.4 Minimum maximal matchings

Finding an ordinary maximal matching in any graph can easily be done by a greedy algorithm in linear time. On the other hand, if we want one with the minimum cardinality, it is not that easy. This problem has been studied by the name of Minimum Maximal Matching (MMM), which consists in finding a maximal matching with the minimum cardinality in a given graph.

Observe a minimum maximal matching, of size 3, in Figure 6a in contrast with a maximum matching, of size 5, on the same graph on Figure 6b.

(a) (b)

Figure 6 – Two maximal matchings, one with minimum cardinality (a) and the other with maximum (b).

This problem has a close relationship with another well-known problem, Edge Dominating Set, in which we are given a graph $G = (V,E)$, and we want to find a dominating edge set having the minimum size, that is, a set $F \subseteq E$ such that every edge of $E \setminus F$ is adjacent to an element of F. The equivalence between these two problems comes from the fact that every maximal matching is an edge-dominating set, and any edge dominating set can be transformed into a maximal matching of the same size in polynomial time [183].

M. Yannakakis and F. Gavril were among the first authors to consider MMM, proving it is NP-hard in bipartite graphs and planar graphs, both with maximum degree 3 [183]. This result was independently improved by an NP-hardness proof for k-regular bipartite graphs for every fixed $k \geq 3$ [41, 42].

Among other negative results in graph classes, we highlight the NP-hardness for planar bipartite graphs and planar cubic graphs [97], and for what was called almost regular bipartite graphs, a subclass of bipartite graphs [189].

On the positive side, MMM has been shown to be solvable in polynomial time for block graphs [100], series-parallel graphs [158], bipartite permutation graphs and cotriangulated graphs [166], and clique-width bounded graphs [57].

There are also extensive studies of approximation algorithms. In 2014, Escoffier et al. showed that the problem is NP-hard to approximate within a factor better than 1.18 [56]. For further results, we refer to [42, 48].

1.5 Stable Marriage

The Stable Marriage (or Stable Matching) is another important variant of the usual matching. In this problem, we are given a bipartite graph $G = (V_1 \dot\cup V_2, E)$ such that $|V_1| = |V_2|$, and an ordering of preferences for each element of the opposite part; we want to find a stable matching, i.e. a matching M such that there is no pair of edges $v_1w_1, v_2w_2 \in M$ such that (i) v_1 prefers w_2 to w_1, and (ii) w_2 prefers v_1 to v_2.

The problem was first solved by D. Gale and L. S. Shapley [72], who showed not only that there always exists a stable matching of size $|V_1| = |V_2|$, but also that it can be obtained in $\mathcal{O}(n^2)$. Besides, one can also seek for optimized stable matchings in terms of preferences of one part over another. That is, given a part V_1, a stable matching such that each vertex in V_1 is matched to the best partner, and each vertex in V_2 to the worst partner, considering all possible stable matchings.

Observe an instance of this problem as a graph $G = (V_1 \dot\cup V_2, E)$, $V_1 = \{A,B,C,D,E\}$, $V_2 = \{a,b,c,d,e\}$ and the preference lists in Table 2. Figure 7 shows two possible stable matchings, one optimizing each bipartition.

There is a good survey of variants that arise from this problem [104]. Some of them are based on (i) incomplete preference lists, where some preference lists may not contain all elements of the other part, (ii) upper bound of stable matchings that a graph admits, or even (iii) optimal stable matchings, where we want to analyze the quality of these matchings; this can be based on some criteria, on what is called *regret cost* $r(M) = \max_{uv \in M}\{\max\{p_{u,v}, p_{v,u}\}\}$, where $p_{u,v}$ denotes the preference of $v \in V_1$ in the list of $u \in V_2$, or even *sex-equalness cost* $d(M) = \sum_{uv \in M}(p_{uv} - p_{vu})$.

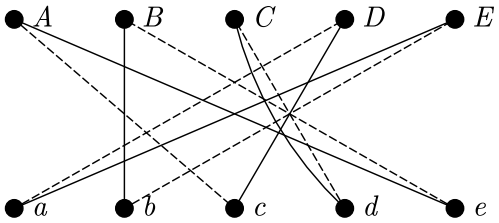


Figure 7 – Two stable matchings, where the dashed is optimal on V_1, while solid, on V_2.

V_1	V_2
$A : cebad$	$a : CBEAD$
$B : ebadc$	$b : ABECD$
$C : dceab$	$c : DCBAE$
$D : abcde$	$d : ACDBE$
$E : bcdae$	$e : ABDEC$

Table 2 – Preference lists of V_1 and V_2.

1.6 Counting all matchings in a graph

In addition to finding a single matching in a graph, the graph theory community is also interested to know the number of matchings that a graph admits.

In 1997, T. Uno showed how to enumerate all maximal, maximum, or perfect matchings in a graph in $\mathcal{O}\big(n^{1/2}m + Nn\big)$ [174] such that N is the number of matchings found. Later, the same author improved the time complexity of counting perfect matchings to $\mathcal{O}\big(n^{1/2}m + N \log n\big)$ [175].

Although these algorithms may perform well, they depend on the number of matchings of the input graph, which can be exponential. This contrasts with the fact that finding a single matching can be done in polynomial time. This difference in counting problems in general motivated L. Valiant to create classes for counting problems in terms of their time complexities, namely #P, #P-complete, FP [177]. In the same paper, L. Valiant showed that the problem of counting all perfect matchings of a bipartite graph is #P-complete, which is considered hard among counting problems.

Later, L. Valiant also proved that it is #P-complete to count matching (not only perfect ones) in bipartite graphs [178]. This result was later strengthened by showing that the problem remains #P-complete for 3-regular bipartite graphs [39]. Other classes for which the problem is #P-complete include chordal and chordal bipartite graphs [144], bipartite graphs of maximum degree 4, and planar bipartite graphs of maximum degree 6 [176].

Interestingly, we can count perfect matchings in polynomial time for planar graphs [112], as well as for chain graphs, cochain, threshold graphs [144], cographs, graphs with bounded treewidth, and complements of chain graphs [49].

1.7 Edge colorings and matchings

Matchings also play an important role in edge colorings. A *(proper) edge coloring* is an edge coloring in which there are no two edges having the same color. A proper edge coloring with k different colors is called a (proper) k-edge-coloring. A graph that admits a k-edge-coloring is said to be k-edge-colorable. The smallest number of colors needed in a (proper) edge coloring of a graph G is the *chromatic index*, or edge chromatic number, $\chi'(G)$. A graph is k-edge-chromatic if its chromatic index is exactly k.

In an edge coloring, the set of edges having the same color must all be non-adjacent two by two, so this set is a matching. Hence, a proper edge coloring is equivalent to a partition of the graph edges into matchings.

One of the most important theorems concerning edge colorings shows that the chromatic index of a graph is either Δ or $\Delta + 1$. In the following, we enunciate this theorem, known as Vizing's Theorem.

Theorem 7 ([179])**.** For a graph G, it holds that $\Delta(G) \leq \chi'(G) \leq \Delta(G) + 1$.

The problem of determining the chromatic index of a graph is NP-hard [95]. However, we know the chromatic index of some graph classes, like bipartite graphs, for which $\chi'(G) = \Delta(G)$ [119].

1.8 Removing matchings from a graph

Another variation of the classic matching problem consists of matchings whose removal result in graphs satisfying certain properties. Two properties studied separately are that the resulting graph is bipartite or acyclic, namely bipartizing and decycling matchings.

Concerning decycling matchings, its problem is NP-complete even for 2-connected planar subcubic graphs [126]. On the other hand, it is solvable in polynomial time for some graph classes, like chordal graphs. Also, it remains NP-complete for Hamiltonian subcubic graphs [156]. On the other hand, for chordal and distance-hereditary graphs the authors presented characterizations of matching-decyclability that lead to linear time recognition algorithms [156].

About bipartizing matchings, it was observed that a graph G admits a bipartizing matching if and only if G has a $(2, 1)$-coloring[2] [127]. The authors showed that the problem is NP-complete for 3-colorable planar graphs of maximum degree 4. On the positive side, they proved that it is solvable in polynomial time in bounded domination

[2] A (k, d)-coloring of a graph G is a k-coloring of $V(G)$ such that each vertex of G has at most d neighbors with the same color as itself.

number graphs, P_5-free graphs, bounded cliquewidth graphs, and graphs in which every odd-cycle subgraph is a triangle.

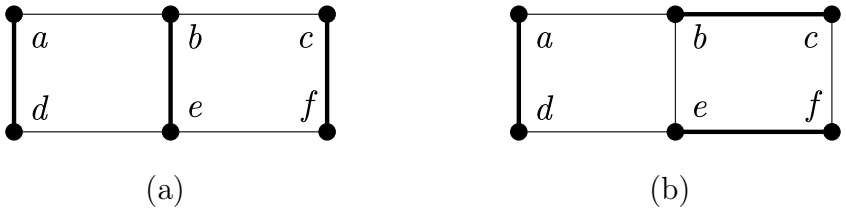
1.9 Line-complete matchings

A matching M is *line-complete* if every two of its edges e_1,$e_2 \in M$ are connected in such a way that $N(e_1) \cap e_2 \neq \emptyset$. This kind of matching was originally named connected matching; we adopt the name line-complete matching in order to avoid the name conflict with connected matching proposed in 2005 [81], described in Section 1.12.

Observe that the matching in Figure 8b is line-complete since all of its edges are connected, that is, (i) $\{a,d\} \cap N(\{b,c\}) = \{a\}$, (ii) $\{a,d\} \cap N(\{e,f\}) = \{d\}$, and (iii) $\{b,c\} \cap N(\{e,f\}) = \{b,c\}$. However, the matching in Figure 8a is not line-complete since $\{a,d\} \cap N(\{c,f\}) = \emptyset$.

Line-complete matchings concept was motivated by a possible solution to Hadwiger's Conjecture [90] and considered in 2003 by Plummer et al. where authors proved the NP-completeness of deciding if a given graph admits a line-complete matching of a given size k [140].

This problem was also approached in the same year by K. Cameron [30], who showed the NP-completeness for its weighted version in bipartite graphs having binary weights. On the positive side, she gave polynomial-time algorithms for chordal graphs and C_4-free graphs. In the same paper, there are characterizations of these matchings in terms of the line graph generated from it and its square graph[3]; in particular, a matching M in a graph G is line-complete if and only if the vertices corresponding to edges of M form an independent set in $L(G)$ and a clique in $[L(G)]^2$. Later, a polynomial-time algorithm for chordal bipartite graphs was designed [108].



(a) (b)

Figure 8 – An ordinary (a) and a line-complete (b) matchings.

1.10 Rank-Maximal Matchings

Another variant consists in, given a bipartite graph, finding a matching where one part (called applicants) has a preference list on the other (posts). Over the years, many optimality criteria were considered, like pareto-optimality [1], popularity [2], fairness [98], and rank-maximality [103]. We give a brief survey about the latter.

Observe in Figure 9a a graph with applicants $\{A,B,C\}$ and their preferences over posts $\{a,b,c\}$. The matching in Figure 9b contains the two 1-rank edges $\{Aa, Bb\}$ and one 3-rank edge $\{Cc\}$.

A rank-maximal matching is one in which the maximum possible number of applicants are matched to their first choice post, and subject to that condition, the maximum possible

[3] A square of a graph G, denoted by G^2, is G plus the edges incident to two of its vertices if their distance in G is 2.

(a) (b)

Figure 9 – A graph with edge ranks in (a), and a rank-maximal matching in (b).

number is matched to their second choice post, and so on. This problem was first studied in 2003 [102], and was motivated by some applications [79], like the allocation of graduates to training positions [101] and families to government housing [185].

In 2006, R. W. Irving et al. presented an $\mathcal{O}(\min\{n+c,c\sqrt{n}\}\cdot m)$ algorithm, where c is the maximal rank of an edge used in rank-maximal matching, n is the number of applicants and posts and m is the total size of the preference lists [103]. Later, the problem was also solved with the same time complexity using a reduction to Maximum Weight Matching [141].

There are other versions concerning rank-maximal matchings, considering vertex weights [113] and capacities [147].

1.11 b-matchings

Another important problem, which is a generalization of the usual matching problem is Maximum b-matching. In this problem, we are given a graph $G = (V,E)$ and a positive integer b_v for each vertex $v \in V$; we want to find a multiset M of E such that, for each $v \in V$, there are at most b_v edges of M incident to v. There is a weighted version of this problem, which consists in finding a b-matching whose sum of its edge weights is maximum. Note that this is a generalization of the classic matching problem if $b_v = 1$ for every $v \in V$.

There are b-matching applications in different domains, which include median location problems [171], spectral data clustering [106] and semisupervised learning [107].

Finding b-matchings of maximum weight is widely studied, having one of its first mentions by Jack Edmonds [52]; it was first solved by a pseudopolynomial algorithm in 1973 by W. R. Pulleyblank [157]. Polynomial algorithms were only designed years later, with the first being presented in 1979 having complexity $\mathcal{O}(n^2 m \log b)$ [131], where $b = \max_{v\in V}\{b_v\}$. In 1983, H. N. Gabow designed a $\mathcal{O}(m^2 \log n \log b)$ algorithm [67] and, then, improved this result with an algorithm having complexity $\mathcal{O}(\min\{B,n\log n\}(m + n\log n))$, where $B = \sum_{v\in V} b_v$ [69].

There are also approaches to the b-matching problem in specific graph classes, such as visibility graphs [171], and distance-hereditary graphs [47]. Recently, the online concept of classical matching was also extended to online b-matching problems [109] [5] [16].

1.12 $\mathscr{P}$-matchings

We finish this survey with a relevant variation that arose over the years, and also the one that we focus on in this thesis: *$\mathscr{P}$-matchings*. Some of their applications include

fields of artificial intelligence [118, 169], game theory [17], computer networks [7, 162], and VLSI design [83].

Let M be a matching of a graph G. We say that M is a $\mathscr{P}$-matching if $G[M]$ satisfies a property $\mathscr{P}$, where $\mathscr{P}$ is a graph property. We denote by $\beta_{\mathscr{P}}(G)$, or by $\mathscr{P}$ matching number, the maximum cardinality of a $\mathscr{P}$-matching. Sometimes, when there is no ambiguity for which graph we are referring to, we may choose to shorten the notation from $\beta_{\mathscr{P}}(G)$ to $\beta_{\mathscr{P}}$. Moreover, we say that a $\mathscr{P}$-matching M is maximal if there is no other $\mathscr{P}$-matching containing M.

Note that, though line-complete matchings of Section 1.9 may be similar to $\mathscr{P}$-matchings, it cannot be classified as one. This is due to the fact that there can be two distinct matchings inducing the same subgraph, and such that only one is line-complete; see an example in Figure 8. Hence, one can not distinguish whether a matching M is line-complete only by its subgraph $G[M]$.

In the following, we summarize some of the properties $\mathscr{P}$ that were considered and studied over the years, for which we give details in the next sections.

- If $\mathscr{P}$ is that of being a general graph. So a $\mathscr{P}$-matching is a usual matching. We denote by $\beta(G)$ the matching number, i.e. the maximum cardinality of a matching that G admits.
- If $\mathscr{P}$ is that the graph is 1-regular (see Figure 10a), we are referring to an *induced matching*, considered in Section 1.12.1.
- A matching M is a *uniquely restricted matching* if $G[M]$ has one unique maximum matching, that is, there is no other matching M' in $G[M]$ such that $|M| = |M'|$ (see Figure 11a). We approach this matching in Section 1.12.2.
- If $\mathscr{P}$ is that the graph is acyclic(see Figure 10b) we say that a $\mathscr{P}$-matching is an *acyclic matching*, which we present in Section 1.12.3.
- We say that a matching M is a *connected matching* if $G[M]$ is connected (see Figure 11b), which we consider in Section 1.12.4.
- A matching M is a *disconnected matching* if $G[M]$ is disconnected (see Figure 12a). Moreover, if $G[M]$ has at least c connected components, then M is *c-disconnected*. We explore disconnected matchings in Section 1.12.5.

Next, we give details of those matchings, and then we compare them in Section 1.12.6, giving bounds on their corresponding matching numbers and some related work. This includes, for example, studying what property a graph G has if it admits an induced matching with the same size as a maximum matching. Finally, in Section 1.12.7, we present the concept of $\mathscr{P}$ edge coloring, which relates $\mathscr{P}$-matchings to edge colorings.

1.12.1 Induced Matchings

A matching M of a graph G is induced or *strong* if $G[M]$ is 1-regular, that is, the edges of M are pairwise not adjacent. We denote by $\beta_*(G)$ the induced matching number, which is the largest cardinality of an induced matching that G admits. Observe in Figure 10a the induced matching $\{hf,cd,aj,ml\}$.

One of the first appearances of this problem was in 1982 [168], when L. J. Stockmeyer and V. V. Vazirani considered δ-separated matching as a restriction of the usual matching.

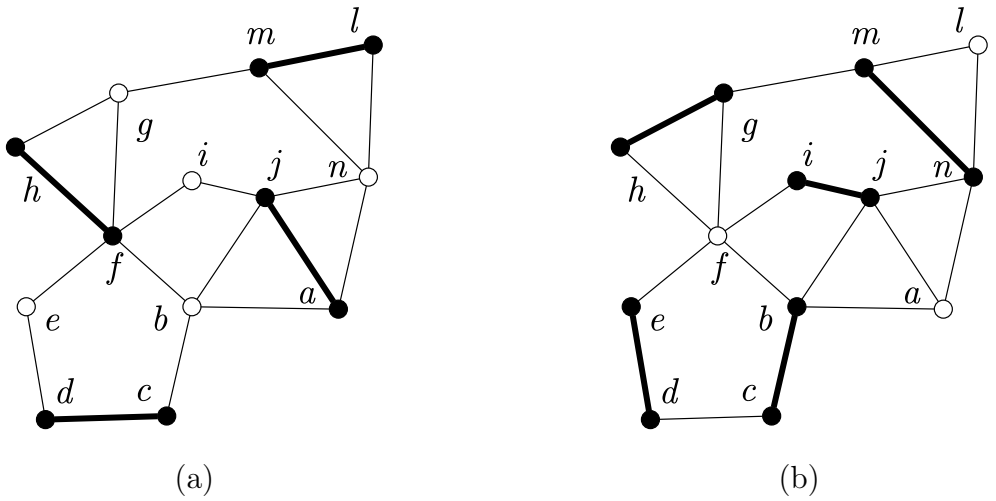


Figure 10 – An induced matching (a) and an acyclic matching (b).

They defined a matching M as δ-separated if, for each pair of edges e_1,$e_2 \in M$, the distance from any vertex of e_1 to any vertex of e_2 is at least δ. Note that any matching is 1-separated. Also, a 2-separated matching is an induced matching and vice-versa. They proved the NP-hardness of finding large δ-separated matchings for bipartite graphs bounded by degree 4, for every fixed $\delta \geq 2$. This implies the NP-hardness in the same conditions of the particular case where $\delta = 2$.

For this specific case, the corresponding problem later received the name of Induced Matching, or Strong Matching. K. Cameron was one of the first to use these terms [28], giving some characterizations and independently proving NP-completeness for bipartite graphs, as well as its pertinence in P for chordal graphs. The latter result makes use of the interesting property that, if G is chordal, then $[L(G)]^2$ is chordal, proved in the same article. It was also based on the observation that every induced matching in G is an independent set in $[L(G)]^2$ and conversely.

This result for chordal graphs was later improved to a linear-time algorithm [23]. Other classes for which the problem can be solved in polynomial time include weakly chordal graphs [31], permutation and trapezoid graphs [26], circular-convex bipartite graphs and triad-convex bipartite [151], asteroidal-triple-free graphs [29, 33], interval-filament graphs [29] and hexagonal graphs [55].

On the negative side, besides the results for bipartite graphs with degree 4 [168], we also highlight the NP-completeness for other graph classes, namely 3-regular planar graphs [116], $4k$-regular graphs for $k \geq 1$ [188], bipartite graphs of maximum degree 3 [129], planar bipartite graphs where each vertex in one partite set has degree 2 and each vertex in the other partite set has degree 3 [114], hamiltonian graphs, claw-free graphs, chair-free graphs, line graphs, and d-regular graphs, for $d \geq 5$ [117], star-convex bipartite graphs and perfect elimination bipartite graphs [151], and d-regular graphs, for each $d \geq 3$ [46]. In this last paper, the authors also proved that the problem is APX-complete under the same constraints.

We know, by definition, that every induced matching is also an ordinary matching. So, it holds that $\beta_*(G) \leq \beta(G)$ for the general case. But there may be graphs where the equality is true, like in a P_5, for example. In 2003, it was presented a polynomial time algorithm which, given a graph G, either it finds a maximum induced matching in G, or it claims that $\beta_*(G) \neq \beta(G)$ [117]. For this reason, it is in P to decide if $\beta_*(G) = \beta(G)$,

for a given graph G. Also, there are cases where every maximum matching of a graph G is also induced. M. Fürst and D. Rautenbach proved that this happens if and only if every component of G is a star or a triangle [63].

Concerning parameterized complexity, Induced Matching was also considered using this approach [91]. Authors showed that it is FPT for (k,c), where c is the number of cycles of length four; and for line graphs for the treewidth parameter k.

Another variation is deciding if a given graph G admits maximal induced matching with cardinality at most k. Orlovich et al. showed that this problem is NP-complete for bi-size matched graphs[146]. Later, the same NP-completeness result was reached, but this time for bipartite graphs with degree 4 [145]. In 2021, J. Chaudhary studied this problem in her Ph.D. thesis [35], proving that it is NP-complete for perfect elimination bipartite graphs, star-convex bipartite graphs, and dually chordal graphs, while solvable in polynomial time for cographs.

The weighted version of induced matching was also the aim of many researchers. It is denoted by Weighted Induced Matching and consists in deciding if an edge-weighted graph G admits an induced matching whose sum of its edge weights is at least k. Orlovich et al. stated that this problem can be solved in polynomial time if the input graph respects the property of some forbidden subgraphs they provide [146]. Other positive results include algorithms that run in linear time for convex bipartite graphs [115] and in polynomial time for circular-convex and triad-convex bipartite graphs [150].

1.12.2 Uniquely Restricted Matchings

A matching M is uniquely restricted if there is no other matching M' such that $V(M) = V(M')$. In other words, $G[M]$ has a unique perfect matching. We denote by $\beta_{ur}(G)$ the uniquely restricted matching number, which is the largest cardinality of a uniquely restricted matching that G admits. Observe in Figure 11a the uniquely restricted matching $\{an,bc,de,ij,gh,ml\}$.

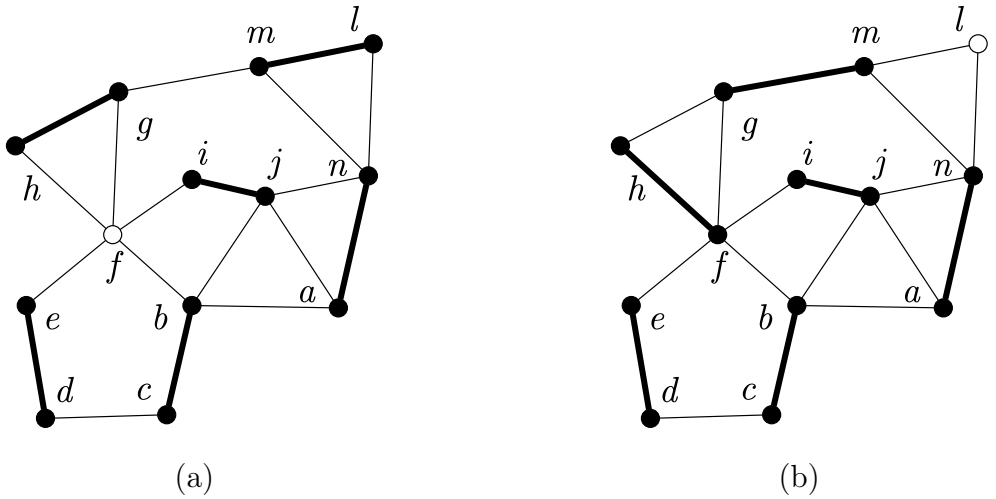


Figure 11 – A uniquely restricted matching (a) and a connected matching (b).

The problem of deciding if a graph admits a uniquely restricted matching of a given size was introduced [82], motivated by a matrix theory problem. In this paper, they gave fundamental characterizations of uniquely restricted matchings. A matching M is uniquely restricted if and only if $G[M]$ does not contain an alternating cycle with respect

to M. Also, they gave an algorithm that tests whether a matching M is uniquely restricted in $\mathcal{O}(|M|m)$. If the graph is bipartite, the run time can be reduced to $\mathcal{O}(n+m)$. This algorithm helps to show that some problems considering uniquely restricted matchings are in NP.

Research on this subject considers the decision problem UNIQUELY RESTRICTED MATCHING, in which we are given a graph G and an integer k, and it is asked if G admits a uniquely restricted matching of size at least k.

This problem is NP-complete for split and bipartite graphs [82]. The latter result was improved to subcubic bipartite graphs, for which it was also proved that the problem is APX-complete [142].

Nevertheless, we can solve it in polynomial time for threshold graphs, proper interval graphs, cacti, block graphs [82], interval, proper interval graphs, bipartite permutation graphs [59].

There is also interest in studying graphs for which the size of a maximum uniquely restricted matching is also the size of a maximum induced matching or of a maximum matching. Note that an induced matching M of a given graph G is uniquely restricted since $G[M]$ is 1-regular. Hence, $\beta_*(G) \leq \beta_{ur}(G) \leq \beta(G)(G)$. However, deciding if $\beta_*(G) = \beta_{ur}(G)$ is NP-hard even for bipartite graphs [62]. If G has no even length cycles, then every matching on G is uniquely restricted, so $\beta_{ur}(G) = \beta(G)$ [82].

Also, J. Chaudhary and B. S. Panda studied the problem where we want to know whether G admits a maximal uniquely restricted matching of cardinality at most k [36]. They showed that this problem is NP-complete for chordal bipartite graphs, star-convex bipartite graphs, chordal graphs, doubly chordal graphs, and even APX-complete for graphs with maximum degree 4. On the positive side, it is solvable in polynomial time for chain graphs [149], bipartite distance-hereditary graphs, bipartite permutation graphs, proper interval graphs, and threshold graphs [36].

1.12.3 Acyclic Matchings

A matching M is acyclic if $G[M]$ is acyclic, that is, a forest. The maximum cardinality of an acyclic matching in a graph G is denoted by $\beta_{ac}(G)$, the acyclic matching number. Observe an acyclic matching $\{bc,de,ij,mn,gh\}$ as example in Figure 10b.

Acyclic matchings were proposed by Goddard et al. in 2005 [81], inspired by induced and uniquely restricted matchings. In the same paper, they showed that it is NP-complete to decide if a graph admits an acyclic matching of size at least k. This result was strengthened as it remains NP-complete for perfect elimination bipartite graphs, combconvex bipartite graphs, dually chordal graphs [148], planar line graphs with maximum degree 4 [91], and perfect elimination bipartite graphs with maximum degree 3 [91].

On the positive side, it is tractable for graphs with a bounded maximum degree [9], chordal graphs [10], interval graphs, split graphs, and proper interval graphs [148].

This problem was also studied from the parameterized point of view [91]. It was proved that it is W[1]-hard for bipartite graphs with respect to the parameter k. However, it is FPT for treewidth parameter and for (k,c), where c is the number of cycles of length four.

Note that, for a graph G, every induced matching is also acyclic, which is also uniquely restricted. So, it holds that $\beta_*(G) \leq \beta_{ac}(G) \leq \beta_{ur}(G) \leq \beta(G)$. Besides, it is hard to decide if $\beta_{ac} = \beta_{ur}$ on general graphs or $\beta_{ac} = \beta_*$ even on graphs with maximum degree 4 [62]. On the other hand, it is known that every maximum matching in G is acyclic if and only if every component of G is a tree or an odd cycle [63].

Goddard et al. also showed that it is NP-hard to compute the minimum cardinality of a maximal acyclic matching [81]. This problem was also studied later for bipartite graphs, perfect elimination bipartite graphs, and proper interval graphs [35].

The weighted version, Weighted Acyclic Matching, was proposed [63], where M. Fürst et al. showed that it is solvable in polynomial time for P_4-free graphs and $2P_3$-free graphs.

The Acyclic Matching problem was later generalized [10], considering r-Degenerate Matching, where we are given a non-negative integer r and a graph G. We want to know if G admits a matching M where $G[M]$ is r-degenerate[4]. Note that a graph is a forest if and only if it is 1-degenerate. For this reason, Acyclic Matching corresponds to 1-Degenerate Matching. In the same article, they described a polynomial time algorithm for determining r-Degenerate Matching in chordal graphs for any fixed r.

1.12.4 Connected Matchings

A matching M is connected if $G[M]$ is connected. The maximum cardinality of a connected matching in a graph G is denoted by $\beta_c(G)$. Observe a connected matching $\{bc,de,fh,gm,ij,an\}$ as example in Figure 11b.

As acyclic matchings, connected matchings were also proposed in 2005 by Goddard et al. [81], where authors showed that $\beta_c(G) = \beta(G)$ (Theorem 2 in [81]). In the proof of this theorem, it is described a procedure that obtains a maximum connected matching from a given maximum matching. Although this procedure is indeed polynomial, it is not clear how efficiently it can be done. In this paper, we clarify this issue by showing a linear algorithm in Section 3.1.

Hence, it is not harder to find maximum connected matchings in comparison to ordinary maximum matchings; in fact, it can be done with the same time complexity of Maximum Matching for general graphs, as discussed in Section 1.1. Additionally, if we restrict to some classes (see Table 1), maximum connected matchings can be obtained even faster than from general graphs.

Although connected matchings were proposed in 2005, there is not much research on them. One of the few papers about it presents an algorithm to compute the minimum cardinality of a maximal connected matching in a tree [105].

In this thesis, we prove we can obtain maximum cardinality connected matchings in linear time if a maximum matching is given. This implies that the complexity of this problem is bounded by the complexity of Maximum Matching.

Besides, we also consider this problem in its edge-weighted version, which we name Weighted Connected Matching; we prove that it is NP-complete for starlike graphs, subcubic planar graphs, planar bipartite graphs, and diameter four bipartite graphs. When we restrict the weights to be non-negative only, it is solvable in polynomial time for chordal graphs; in contrast with this fact, when the weights are arbitrary, it is NP-complete for starlike graphs, a subclass of chordal graphs. It can be solved in linear time for graphs with maximum degree at most 2 and for trees, and in FPT time when parameterized by treewidth. We summarize these results in Table 3.

One interesting approach that has not yet been studied is minimum maximal connected matchings. Note that, unlike the equality $\beta(G) = \beta_c(G)$ holds for maximum matchings,

[4] A graph G is r-degenerate if every subgraph H of G of order at least 1 has a vertex of degree at most r in H

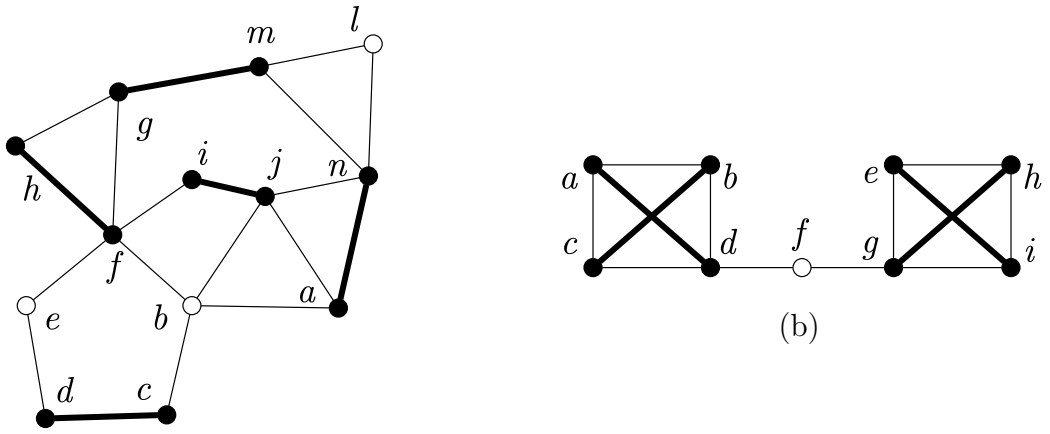
Table 3 – Summary of our results for Weighted Connected Matching.

<table>
<tr><td colspan="2" rowspan="2">Graph class</td><td colspan="2">Complexity</td></tr>
<tr><td>Weights ≥ 0</td><td>Any weights</td></tr>
<tr><td colspan="2">General</td><td colspan="2" rowspan="2">NP-complete
(Theorem 23)</td></tr>
<tr><td colspan="2">Bipartite having
diameter at most 4</td></tr>
<tr><td colspan="2">Chordal</td><td rowspan="2">P
(Theorem 22)</td><td rowspan="2">NP-complete
(Theorem 21)</td></tr>
<tr><td colspan="2">Starlike</td></tr>
<tr><td rowspan="2">Planar</td><td>bipartite</td><td colspan="2">NP-complete
(Theorem 24)</td></tr>
<tr><td>subcubic</td><td>?</td><td>NP-complete
(Theorem 25)</td></tr>
<tr><td colspan="2">$\Delta \leq 2$</td><td colspan="2">P
(Theorem 26)</td></tr>
<tr><td colspan="2">Tree</td><td colspan="2">P
(Theorem 29)</td></tr>
</table>

it is not known to hold for minimum maximal matchings. Indeed, a minimum maximal connected matching can be bigger than a minimum maximal matching. For example, in C_6, the size of the former is 3 while the size of the latter is 2.

1.12.5 Disconnected Matchings

A matching M is called c-disconnected if $G[M]$ has at least c connected components. We denote the cardinality of a maximum i-disconnected matching in a graph G by $\beta_{d,i}(G)$. When $c = 2$, we can simply say that M is disconnected and $\beta_{d,2}(G) = \beta_d(G)$. Observe in Figure 12a the disconnected matching $\{hf,cd,ij,an,mg\}$, and a 4-disconnected matching $\{hf,cd,ja,ml\}$ in Figure 10a.



(a) (b)

Figure 12 – A disconnected matching (a) and a graph that has the disconnected matching number larger than uniquely restricted and acyclic matching numbers (b).

The problem of deciding if a graph admits a disconnected matching of a given size

was first proposed in 2005 [81], where Goddard et al. asked about its complexity. In this thesis, we answer this question, showing it is NP-complete even for bipartite graphs.

In 2007, A. Jamieson presented a linear-time algorithm that obtains both the maximum cardinality as well as the minimum cardinality of a maximal disconnected matching for a given tree [105]. Later, two algorithms were given to find the maximum disconnected matchings, one that runs in linear time for block graphs, and the other in $\mathcal{O}(mn)$ for chordal graphs [133].

In this paper, we go further, generalizing the problem considering the desired number of connected components c as a constant or as part of the input of the problem. We assert that for every fixed $c \geq 2$, Disconnected Matching remains NP-complete for bipartite graphs, but can be solved in XP time for the parameter c in graphs with a polynomial number of minimal separators. This includes chordal graphs and, consequently, their subclasses, like interval graphs.

On the other hand, if we put c as part of the input, the problem turns out to be NP-complete on chordal graphs, while it is still solvable in polynomial time for interval graphs.

We summarize these results in Table 4.

Table 4 – Complexity results for Disconnected Matching restricted to some input scopes. We denote by η_i the i-th Bell number.

Graph class	c	Complexity	Proof
General	$c = 1$	Same as Maximum Matching	Theorem 2
Bipartite	Fixed $c \geq 2$	NP-complete	Theorem 9
Chordal	Input	XP and NP-complete	Theorems 11 and 13
Bounded degree	Input	NP-complete	Theorem 11
Interval	Input	$\mathcal{O}(n^2 c \max\{nc, m\sqrt{n}\})$	Theorem 14
Treewidth t	Input	$\mathcal{O}\big(8^t \eta_{t+1}^3 n^2\big)$	Theorem 15

We can see that, for a given graph G, it holds that $\beta(G) = \beta_{d,1}(G) \geq \beta_{d,2}(G) \geq \beta_{d,3}(G) \geq \ldots \geq \beta_{d,\beta_*}(G) \geq \beta_*(G)$. Note that $\beta_*(G)$ may be strictly smaller than $\beta_{d,\beta_*}(G)$, as the in the graph of Figure 12b. But one may ask if we can determine if $\beta_{d,i}(G) = \beta_{d,j}(G)$. We show that this can not be done in polynomial time unless P = NP.

Concerning parameterized complexity, it was given an FPT algorithm for the treewidth parameter [88].

Like connected matchings, the subject of minimum maximal disconnected matchings has not been studied yet. The sizes of a minimum maximal disconnected and a minimum maximal matching may be the same (in a P_5) or not (Figure 6). We leave as an open problem to decide if a given graph admits a maximal disconnected matching with at most k edges.

Concerning the weighted version of this matching, it has not been considered yet. We also would like to know if there are graph classes for which Weighted Disconnected Matching can be solved in polynomial time.

1.12.6 $\mathscr{P}$-matching numbers

One can easily see that every induced matching is also disconnected, acyclic, and uniquely restricted. Besides, every acyclic matching is also uniquely restricted; every i-disconnected matching is a j-disconnected matching, for $j > i$.

For this reason, for a given graph G, we have the following chains of inequalities.

$$\beta_*(G) \leq \beta_{ac}(G) \leq \beta_{ur}(G) \leq \beta(G) \tag{1.1}$$

$$\beta_*(G) \leq \beta_{d,\beta_*}(G) \leq \ldots \leq \beta_{d,3}(G) \leq \beta_{d,2}(G) \leq \beta_{d,1}(G) \tag{1.2}$$

Even though the induced matching number is a lower bound for all the other $\mathscr{P}$-matchings mentioned, there is no relation between some of them. For instance, the disconnected matching number is not comparable to uniquely restricted or acyclic matching numbers for general graphs. Note that in the graph of Figure 11a, $6 = \beta_{ur} > \beta_d = 5$, while in Figure 12b, $3 = \beta_{ur} < \beta_d = 4$. Concerning acyclic matchings, note that in a P_6, $3 = \beta_{ac} > \beta_d = 2$, while in Figure 12b, $2 = \beta_{ac} < \beta_d = 4$.

By equations (1.1) and (1.2), it seems natural to ask in what conditions the equality holds between two $\mathscr{P}$-matching numbers. Many of these comparisons were made over the years. We present some in Table 5.

Table 5 – Complexity of deciding if a graph has equal $\mathscr{P}$-matching numbers.

Problem	Complexity	Graph class	Proof
$\beta_c = \beta$	P	General	[81] Theorem 19
$\beta_* = \beta$	P	General	[117]
$\beta_* = \beta_{ur}$	NP-hard	Bipartite	[61]
$\beta_{ur} = \beta_{ac}$	NP-hard	General	[63]
$\beta_* = \beta_{ac}$	NP-hard	$\Delta = 4$	[61]
$\beta = \beta_{ac}$	NP-complete	$\Delta = 4$	[63]
$\beta_{d,i} = \beta_{d,j}$	NP-hard	Bipartite	Theorem 18
$\beta_d = \beta$	NP-complete	Bipartite	Corollary 6

It is worth mentioning that the complexity for $\beta_* = \beta_d$ has recently been determined by Guilherme C. M. Gomes, Bruno P. Masquio, Paulo E. D. Pinto, Dieter Rautenbach, Vinicius F. dos Santos, Jayme L. Szwarcfiter, and Florian Werner. A preliminary technical report will be published soon. In Section 2.5, we present partial results of this work, which shows that the problem $\beta_{d,i} = \beta_{d,j}$ is NP-hard.

1.12.7 $\mathscr{P}$-matchings and Edge Coloring

The edge coloring concept was extended and related to $\mathscr{P}$-matchings. An edge coloring is said to be a $\mathscr{P}$ edge coloring if every set of edges having the same color is a $\mathscr{P}$-matching. Moreover, $\chi'_{\mathscr{P}}(G)$, denoted by $\mathscr{P}$ chromatic index, is the minimum number of colors needed for a $\mathscr{P}$ edge coloring in a graph G.

Some of the properties $\mathscr{P}$ studied are strong (induced), uniquely restricted [11] and r-degenerate [10] edge colorings.

The problem of strong edge coloring was proven to be NP-complete even for bipartite graphs with a girth of at least four [130]. On the other hand, a strong edge coloring of G is the proper vertex coloring of the square of the line graph of G [130]. Hence, it is polynomially solvable for chordal graphs [28], co-comparability graphs [83], and partial k-trees [160].

In the following, we present an important conjecture about an upper bound on the strong chromatic index.

Conjecture 1 ([54, 93, 181])**.**

$$\chi'_s(G) \leq \begin{cases} \frac{5}{4}\Delta(G)^2 & \text{if } \Delta \text{ is even} \\ \frac{5}{4}\Delta(G)^2 - \frac{1}{2}\Delta(G) + \frac{1}{4} & \text{if } \Delta(G) \text{ is odd} \end{cases}$$

Although this conjecture is not yet solved, there are some upper bound results, showing that $\chi'_s(G) \leq 1.93\Delta(G)^2$ [27], which was improved to $\chi'_s(G) \leq 1.772\Delta(G)^2$ [99]. For further results in terms of $\chi'_s(G)$, we refer to the surveys presented [182, 187].

Degenerate chromatic index, denoted by $\chi'_r(G)$, was studied in 2018 [10], where it was showed that $\chi'_s(G) \geq \chi'_1(G) \geq \chi'_{ur}(G) \geq \chi'(G)$. Also, let $r > 0$ and G such that its maximum degree is at most $\triangle$. It holds that $\chi'_r(G) \leq \frac{2(\Delta-1)^2}{r+1} + 2(\Delta - 1) + 1$. Moreover, $\chi'_1(G) = \Delta^2$ if and only if G is isomorphic to $K_{\Delta,\Delta}$.

Edge coloring considering uniquely restricted matchings, whose chromatic index is denoted by $\chi'_{ur}(G)$, was approached in [11]. In this paper, the authors provided the following tight upper bounds in terms of the maximum degree and characterize all extremal graphs. Let G be a connected graph of maximum degree at most Δ, then $\chi'_{ur}(G) \leq \Delta^2$ with equality if and only if G is isomorphic to $K_{\Delta,\Delta}$. Moreover, if $\Delta \geq 4$, not isomorphic to $K_{\Delta,\Delta}$, then $\chi'_{ur}(G) \leq \Delta^2 - \Delta$.

Since matchings are highly related to edge colorings, it would also be interesting to study other $\mathscr{P}$ edge colorings besides strong, uniquely restricted and degenerate. For instance, one could obtain interesting characterizations for acylcic, connected and disconnected edge colorings.

2 DISCONNECTED MATCHINGS

In this chapter, we study the $\mathscr{P}$-matching for the property of being disconnected. A c-disconnected matching M is a matching such that $G[M]$ has at least c connected components. We will focus on the decision problem of finding disconnected matchings of large cardinality, which is defined as follows.

DISCONNECTED MATCHING
Instance: A graph G and two integers k and c.
Question: Is there a matching M with at least k edges such that $G[M]$ has at least c connected components?

We also approach this problem for a fixed c; we denote this version by c-DISCONNECTED MATCHING.

The concept of disconnected matchings was proposed in 2005 [81], where an exponential algorithm for finding ones with large cardinalities was presented, which we describe next. First, observe that a disconnected matching in a connected graph contains at least one non-saturated minimal separator. For each minimal separator S, we can run the MAXIMUM MATCHING algorithm in $G - S$. The largest matching found is also a maximum disconnected matching in G.

This algorithm is indeed exponential since an arbitrary graph may have an exponential number of minimal separators. On the other hand, we can use it to find maximum disconnected matchings on graphs that all minimal separators can be enumerated in polynomial time. There are some classes, like chordal graphs, for which this property holds [163].

As a specific case of this property, Goddard et al. considered the problem for a tree T, showing that $\beta(T) - 1 \leq \beta_d(T) \leq \beta(T)$ [81]. The proof relies on the fact that every minimal separator in T is a single vertex. So, the previous algorithm runs in polynomial time.

Goddard et al. asked about the complexity of calculating $\beta_d(G)$ of a given graph G [81]. In this paper, we answer this question, showing that c-DISCONNECTED MATCHING is NP-complete for $c = 2$. We also go further, proving that it remains NP-complete even for bipartite graphs having bounded diameter and for every fixed $c \geq 2$.

Moreover, we found out that when c is part of the input, the problem turns out to be NP-complete for bounded degree graphs and chordal graphs. This contrasts with the fact that, for a fixed c, as described previously, we can find a maximum disconnected matching in polynomial time for chordal graphs. Despite this, for interval graphs, a subclass of chordal graphs, we show that DISCONNECTED MATCHING can be solved in polynomial time.

There is also a strong relationship between DISCONNECTED MATCHING and INDUCED MATCHING. Observe that, if we can find a k-disconnected matching M then we can easily obtain an induced matching $M^* \subseteq M$ with k edges. This immediately results in the following theorem, concerning the complexity of DISCONNECTED MATCHING.

Theorem 8. Disconnected Matching is NP-complete for every graph class for which Induced Matching is NP-complete.

Proof. Note that for every input instance (G,k) of Induced Matching, we can build an equivalent instance (G,k,k) of Disconnected Matching. That is, we want to find, in the same graph G, a disconnected matching M with at least k edges and k connected components. To obtain the induced matching, it suffices to pick, for each connected component of $G[M]$, exactly one edge. Finally, observe that an Induced Matching on k edges is also a k-disconnected matching with k edges. □

Next, we summarize a collection of immediate observations concerning induced, disconnected and ordinary matchings.

Observation 1. Let G be a connected graph. These statements hold:

1. Every matching is a 1-disconnected matching.
2. Since $\beta_*(G)$ is the maximum number of components that $G[M]$ can have with any matching M, there exists no c-disconnected matching for $c > \beta_*(G)$.
3. Every maximum induced matching M^* is a $\beta_*(G)$-disconnected matching since each connected component of $G[M^*]$ is an edge.

Moreover, as we mentioned in Equation 1 in Section 1.12.6 the following chain of inequalities is valid:

$$\beta = \beta_{d,1} \geq \beta_{d,2} \geq \beta_{d,3} \geq \ldots \geq \beta_{d,\beta_*} \geq \beta_*$$

Note that, the two last matching numbers of this chain can be equal or not. For instance, $\beta_{d,\beta_*} = \beta_*$ in a P_5, while $\beta_{d,\beta_*} > \beta_*$ in the graph of Figure 12b.

The rest of this chapter is organized as follows. In Section 2.1, we present a construction used to show that c-Disconnected Matching is NP-complete for every fixed $c \geq 2$ on bipartite graphs of diameter three. In Section 2.2, we prove some negative results, that Disconnected Matching is NP-complete on chordal graphs and bounded degree graphs. We show, in Section 2.3.1, that the previous proof cannot be strengthened to fixed c by giving an XP algorithm for Disconnected Matching parameterized by c on graphs with a polynomial number of minimal separators. In Sections 2.3.2 and 2.3.3, we mention polynomial-time algorithms for Disconnected Matching in interval and bounded treewidth graphs. In Section 3.2.5, we approach the problem from the kernelization point of view, in a way that Induced Matching problem, and consequently Disconnected Matching, does not admit a polynomial kernel when parameterized by vertex cover and size of the matching nor when parameterized by vertex deletion distance to clique and size of the matching. Finally, in Section 2.5, we show that it is NP-hard to decide, for a given graph, if $\beta_{d,i}(G) = \beta_{d,j}(G)$, for every $1 \leq i < j$.

2.1 Complexity of c-Disconnected Matching

2.1.1 1-disconnected matchings

We consider that the input graph has at least one edge and is connected. Otherwise, the solution is trivial or we can solve the problem independently for each connected component. Recall that 1-Disconnected Matching allows its solution to have any

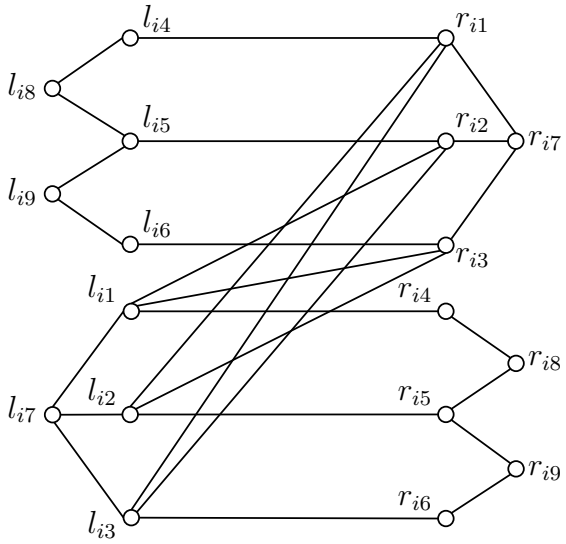


Figure 13 – The subgraph B_i, related to clause c_i

number of connected components. Consequently, any matching with at least k edges is a valid solution to an instance $(G, k, 1)$, which leads to our next observation.

Observation 2. 1-Disconnected Matching is equivalent to Maximum Matching.

2.1.2 2-disconnected matchings

Next, we show that 2-Disconnected Matching is NP-complete for bipartite graphs with bounded diameter. Our reduction is from the NP-hard problem One-in-three 3SAT [78]. In this problem, we are given a set of m clauses I with exactly three literals in each clause, and ask if there is a truth assignment of the variables such that only one literal of each clause resolves to true. We consider that each variable must be present in at least one clause and that a variable is not repeated in the same clause. These constraints follow from the original NP-completeness reduction [161].

2.1.2.1 Input transformation in One-in-three 3SAT.

We use $k = 12m$ and build a bipartite graph $G = (V_1 \dot{\cup} V_2, E)$ from a set of clauses I according to the following construction rules number I to V.

(I) For each clause c_i, generate a subgraph B_i as described below.

- $V(B_i) = \{l_{ij}, r_{ij} \mid j \in \{1, \ldots, 9\}\}$
- $E(B_i)$ is as shown in Figure 13.

(II) For each variable x present in two clauses c_i and c_j, being the q-th literal of c_i and the t-th literal of c_j, add two edges. If x is negated in exactly one of the clauses, add the set of edges $\{r_{iq}l_{jt}, l_{i(q+3)}r_{j(t+3)}\}$. Otherwise, add $\{l_{i(q+3)}r_{jt}, r_{iq}l_{j(t+3)}\}$.

- As an example, consider $c_2 = (x \vee y \vee \overline{z})$ and $c_5 = (d \vee \overline{x} \vee \overline{g})$. Variable x is present in c_2 as the first literal and, in c_5, as the second literal. Besides, x is negated only in c_5. Hence, we add the set of edges $\{(r_{21}l_{52}), (l_{24}r_{55})\}$. In Figure 14, we present this example but omit some edges for better visualization.

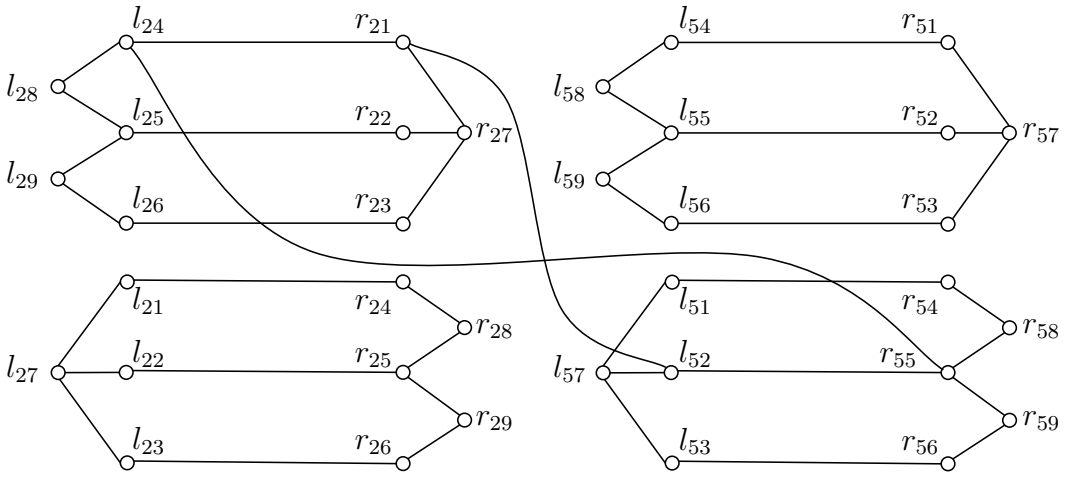


Figure 14 – The simplified subgraph $G[V(B_2) \cup V(B_5)]$ for an instance with the clauses $c_2 = (x \vee y \vee \overline{z})$ e $c_5 = (d \vee \overline{x} \vee \overline{g})$

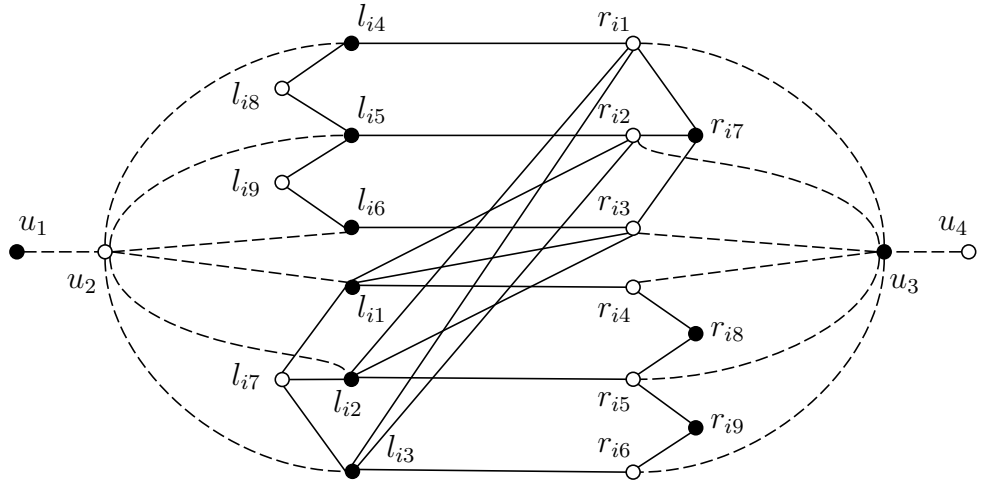


Figure 15 – The subgraph $G[V(B_i) \cup \{u_1, u_2, u_3, u_4\}]$, $u_1 \in V(U_1)$, $u_2 \in V(U_2)$, $u_3 \in V(U_3)$ and $u_4 \in V(U_4)$. The bold vertices represent a bipartition of G.

(III) Generate two complete bipartite subgraphs H_1 and H_2, both isomorphic to $K_{3m,3m}$, $V(H_1) = V(U_1) \dot{\cup} V(U_2)$ and $V(H_2) = V(U_3) \dot{\cup} V(U_4)$.

(IV) For each $u_2 \in V(U_2)$ and clause c_i, add the set of edges $\{u_2 l_{ij} \mid j \in \{1, \ldots, 6\}\}$.

(V) For each $u_3 \in V(U_3)$ and clause c_i, add the set of edges $\{u_3 r_{ij} \mid j \in \{1, \ldots, 6\}\}$.

Besides G being bipartite, as shown in Figure 15, it is possible to observe that its diameter is 5, regardless of the set of clauses and its cardinality. This holds due to the distance between, for example, u_1 and u_4, $u_1 \in V(U_1)$, $u_4 \in V(U_4)$, as well as l_{i7} and r_{j7}, for all distinct $i,j \in [m]$, such that the clauses c_i and c_j do not have literals related to the same variable. Also, consider $G_i^+ = G[V(B_i) \cup V(H_1) \cup V(H_2)]$. Note that $|V(G)| = \mathcal{O}(m)$.

We denote A by the subgraph induced by the vertices of $V(G) \setminus V(H_1) \setminus V(H_2)$. Note that A is exactly the subgraph induced by the vertex set of $\cup_{i \in [m]} V(B_i)$.

Observe that $|V(G)| = 30m$, since $V(G) = V(H_1) \cup V(H_2) \cup V(A)$. Besides, $(45m^2 + 26m) \leq |E(G)| \leq (48m^2 + 23m)$, as the following amounts of edges are generated in the construction. There are $26m$ edges introduced by rule (I), $9m^2$ by (III), and $36m^2$ by (IV) and (V) combined. Rule (II) produces a variable number of edges, which can range from

0 to $3m^2 - 3m$, since each pair of clause subgraphs can have from 0 to 6 edges between them.

2.1.2.2 Properties of disconnected matchings in the generated graphs

We now prove some properties of the disconnected matching with cardinality at least k in a graph G generated by the construction previously described.

Initially, we show, by Lemmas 1 and 2, that a subgraph induced by the saturated vertices of such matching has exactly two connected components, one containing vertices of H_1 and the other, vertices of H_2. Then, Lemma 3 shows the sets of edges that can possibly be in the matching.

Lemma 1. If M is a disconnected matching with cardinality $k \geq 12m$, then there exists two saturated vertices $h_1 \in V(H_1)$ and $h_2 \in V(H_2)$.

Proof. In order to obtain M with cardinality k, it is necessary that $2k \geq 2 \cdot (12m) = 24m$ vertices are saturated by M. Note that $|V(A)| = 18m$. Since we are looking for a k cardinality matching, then, even if all the vertices of A were saturated, we would have, at most, $18m$ vertices. Therefore, for M to saturate $2k$ vertices, we need to use vertices of $V(H_1) \cup V(H_2)$. Note that, similarly, if vertices of A and only one of the subgraphs H_1 or H_2, $|V(H_1)| = |V(H_2)| = 6m$, we will have a maximum of $24m$ vertices, however, the matching would be perfect in the subgraphs $G - V(H_1)$ or $G - V(H_2)$ and, therefore, connected. Thus, it is necessary that there are at least two vertices $h_1 \in V(H_1)$ and $h_2 \in V(H_2)$ saturated by M. □

Lemma 2. If M is a disconnected matching with cardinality $k \geq 12m$, then $G[M]$ has exactly two connected components.

Proof. From Lemma 1, we know that M saturates $h_1 \in V(H_1)$ and $h_2 \in V(H_2)$ by two edges, (h_1, v_1) and (h_2, v_2). Note that, due to the graph structure, every edge e saturated by M is incident to at least one vertex of $V(G) \setminus \{l_{i7}, l_{i8}, l_{i9}, r_{i7}, r_{i8}, r_{i9} \mid i \in [m]\}$, as every edge of the graph has this property. Then, one end of e is adjacent to any of the vertices in $\{h_1, v_1, h_2, v_2\}$. Therefore, $G[M]$ has exactly two connected components C_1 and C_2 such that $h_1 \in V(C_1)$ and $h_2 \in V(C_2)$. □

Lemma 3. Let M be a disconnected matching with cardinality $k \geq 12m$ and B_i be a clause subgraph. There are exactly 6 edges in M of $G[V(B_i)]$ and, moreover, there are exactly 3 sets of edges that satisfy this constraint.

Proof. Let M be a 2-disconnected matching in G, $|M| \geq 12m$, and B_i be a clause subgraph. From Lemma 1, we know that there are two saturated vertices $u_2 \in V(U_1)$ and $u_3 \in V(U_3)$. Also, Lemma 2 shows that any other saturated vertex in the graph must be in one of the two connected components of $G[M]$ containing u_2 or u_3. Therefore, there is a $u_2 - u_3$ separator S_i not saturated in B_i.

For the rest of the proof, we use S_i separators with cardinality 6, so that all the 12 vertices of $V(B_i) \setminus S_i$ will be saturated by M. We prove that there are only 3 separators of this type, due to the following properties.

1. The vertex pairs l_{ij} and $r_{i(j+3)}$ cannot be saturated simultaneously. Thus, S_i contains at least one of these two vertices.

2. If there are two saturated vertices l_{ij} and l_{iq}, then the four vertices r_{ij}, r_{iq}, r_{it} and r_{i7} cannot be saturated, for distinct $j,q,t \in \{1,2,3\}$.

3. The vertex l_{ij} cannot be saturated simultaneously with r_{iq} or r_{it}, for all distinct $j,q,t \in \{1,2,3\}$.

4. The number of saturated vertices of $\{l_{i8}, l_{i9}\}$ must be at most the number of saturated vertices of $\{l_{i4},l_{i5},l_{i6}\}$.

Consider W_i the set of vertices in $V(B_i) \setminus S_i$ that could possibly be saturated by M. From Property 1, we can see that $|S_i| \geq 6$. Thereby, $|W_i| \leq 12$, that is, the largest number of saturated vertices in a clause subgraph is 12. Next, we will show that there are only 3 separators and W_i sets of that type. If $|S_i| = 6$, which is its minimum cardinality, then, given Property 2, there can only be a single saturated vertex l_{ij}, $j \in \{1,2,3\}$. In addition, for r_{i7} to be in W_i, a vertex r_{it}, $t \in \{1,2,3\}$ must be in W_i as well. Given Property 3, the only possibility is if $t = j$. Thus, the vertices $l_{i(j+3)}$ and $r_{i(j+3)}$ cannot belong to W_i. In addition, from Property 4, for l_{i8} and l_{i9} to be in W_i, then two vertices $l_{i(q+3)}$ are in W_i as well, for two distinct $q \in \{1,2,3\}$. The only possibility of this occurring is if $q \neq j$. Analogously, the same is true for the vertices $r_{i(q+3)}$. Finally, we can define the set described as $W_i = \{l_{ij}, r_{ij}, l_{i(q+3)}, r_{i(q+3)}, l_{i(t+3)}, r_{i(t+3)} \mid j,q,t \in \{1,2,3\}, \text{distinct}\} \cup \{l_{ij}, r_{ij} \mid j \in \{7,8,9\}\}$. Therefore, there are only 3 possibilities for the set W_i, which are shown in Table 6 and in Figure 16. Moreover, there is exactly one corresponding saturated set of edges for each of the vertex sets, shown in Figure 16. □

2.1.2.3 Transforming a disconnected matching into a variable assignment.

First, we define, starting from a 2-disconnected matching M, $|M| = 12m$, a variable assignment R and, in sequence, we present Lemma 4, proving that R is a One-in-three 3SAT solution.

(I) For each clause c_i, where x_{ij} corresponds to the j-th literal of c_i, generate the following assignments.

- If l_{ij} is M-saturated, then assign $x_{ij} = T$.
- Otherwise, assign $x_{ij} = F$.

Note that, analyzing the generated graph, the pair of saturated vertices l_{ij} and r_{ij}, $j \in \{1,2,3\}$ define that the j-th literal is the true of the clause c_i. Similarly, each pair of saturated vertices l_{iq} and r_{iq}, $q \in \{4,5,6\}$, $q \neq j+3$, defines that the $(q-3)$-th literal is false.

Lemma 4. Let M be a 2-disconnected matching with cardinality $k = 12m$ in a graph generated from a input I of One-in-three 3SAT. It is possible to generate in polynomial time an assignment to variables in I that solves One-in-three 3SAT.

Proof. For this lemma hold, using the R assignments, each clause in I must have exactly one true literal and each variable must have the same assignment in all clauses. As it was deduced in the Lemma 3, in fact, given $i \in [m]$, l_{ij} is saturated only for a single j, $j \in \{1,2,3\}$. So we have a single true literal. We now show that the assignment of the variable is consistent across all clauses. By contradiction, assume this to be false,

$h_1 - h_2$ separator of G_i^+	Possibly saturated remaining vertices
$\{l_{i2}, l_{i3}, l_{i4}, r_{i2}, r_{i3}, r_{i4}\}$	$\{l_{i1}, l_{i5}, l_{i6}, l_{i7}, l_{i8}, l_{i9}, r_{i1}, r_{i5}, r_{i6}, r_{i7}, r_{i8}, r_{i9}\}$
$\{l_{i1}, l_{i3}, l_{i5}, r_{i1}, r_{i3}, r_{i5}\}$	$\{l_{i2}, l_{i4}, l_{i6}, l_{i7}, l_{i8}, l_{i9}, r_{i2}, r_{i4}, r_{i6}, r_{i7}, r_{i8}, r_{i9}\}$
$\{l_{i1}, l_{i2}, l_{i6}, r_{i1}, r_{i2}, r_{i6}\}$	$\{l_{i3}, l_{i4}, l_{i5}, l_{i7}, l_{i8}, l_{i9}, r_{i3}, r_{i4}, r_{i5}, r_{i7}, r_{i8}, r_{i9}\}$

Table 6 – Some $h_1 - h_2$ minimal separators in a subgraph G_i^+ and the respective remaining sets of vertices, which can be saturated

then in R there are two literals, x and y, for the same variable, and one of the following two possibilities occurs. Consider x the q-th literal of c_i and y the t-th literal of c_j. Either there is one negation between x and y or x and y have the same sign. In the first possibility, as we assumed that x and y have different assignments, then either r_{iq} and l_{jt} are saturated simultaneously or $l_{i(q+3)}$ and $r_{j(t+3)}$ are. Note that c_i and c_j have variables with opposite literals, which means that the constructed graph has the edges $r_{iq}l_{jt}$ and $l_{i(q+3)}r_{j(t+3)}$. Therefore, by Lemma 1, $G[M]$ would be connected, which is a contradiction. In the second possibility, x and y have the same sign. So either $l_{i(q+3)}$ and r_{jt} are saturated simultaneously or r_{iq} and $l_{j(t+3)}$ are. There are also edges between these pairs of vertices and, also by Lemma 1, it is a contradiction. Therefore, R solves One-in-three 3SAT. □

2.1.2.4 Transforming a variable assignment into a disconnected matching.

Finally, we define a 2-disconnected matching M, obtained from a solution of One-in-three 3SAT. Then, Lemma 5 proves that M is a 2-disconnected matching with the desired cardinality $12m$.

(I) For each clause c_i, whose true literal is the j-th, add to M the set of edges defined as $\{l_{ij}l_{i7}, r_{ij}r_{i7}, l_{iq}l_{i8}, r_{iq}r_{i8}, r_{it}l_{i9}, r_{it}r_{i9} \mid q \in \{4,5\}, t \in \{5,6\}, q \neq j+3 \neq t \neq q\}$.

(II) For H_1, add to the matching M any $3m$ disjoint edges. Repeat the process for H_2.

Lemma 5. Let R be a variable assignment of an input I from One-in-three 3SAT. It is possible, in polynomial time, to generate a disconnected matching with cardinality $k = 12m$ from I in a graph generated by the transformation described below.

Proof. In the procedure described, we are saturating 6 edges for each clause in (I), and $6m$ edges in (II). Then, M has $12m$ edges. We need to show now that M is disconnected. It is necessary and sufficient to show that there are no M-saturated two adjacent vertices l and r. Edges between vertices of the same clause subgraph are generated in (I) and we observe that there are no two adjacent M-saturated vertices of this type, since the M-saturated vertices are those described in Lemma 3. The vertices incident to the edges between different clause subgraphs cannot be simultaneously M-saturated, since a variable cannot be true and false at the same time. Therefore, M is disconnected and $|M| = 12m$. □

Note that for any graph with diameter $d \leq 1$ the answer to Disconnected Matching is always NO. On the other hand, if the graph is disconnected, there are two possibilities. If the graph has no more than one connected component with more than one vertex, we again answer NO. Otherwise, the problem can be solved in polynomial time by finding a

maximum matching M and checking if $|M| \geq k$. These statements are used in the proof of Lemma 6, which has a slight modification of the above construction, but allows us to reduce the diameter of the graph to 3.

Lemma 6. Let $G = (V_1 \dot{\cup} V_2, E)$ be the bipartite graph from the transformation mentioned and $G' = (V_1' \dot{\cup} V_2', E')$ so that $V(G') = V(G) \cup \{w_1, w_2\}$ and $E(G') = E(G) \cup \{w_1 w_2\} \cup \{vw_1 \mid v \in V(V_1)\} \cup \{vw_2 \mid v \in V(V_2)\}$. If M is a 2-disconnected matching in G', $|M| \geq k$ and M is also a 2-disconnected matching in G.

Proof. Let us show that a 2-disconnected matching M in G', $|M| \geq 12m$, saturates only vertices of $V(G)$ and, therefore, M is also a 2-disconnected matching in G. With this purpose, we demonstrate that the vertices w_1 and w_2 are not part of M. Let us assume that w_1 is saturated and $w_1 \in C_1$. Note that, since G' is bipartite, then every edge $e \in E(G')$ has one endpoint at V_1' and other at V_2'. Therefore, the edge e, if saturated, would be at C_1. Thereby, M would not be 2-disconnected, which is a contradiction. This shows that w_1 is not saturated. The argument is analogous to w_2. Thus, if M is a 2-disconnected matching in G', $|M| \geq 12m$, so it is also in G. As we have already described the structure of such matchings in G in Lemmas 1, 2 and 3, this transformation can also be used to solve the One-in-three 3SAT problem. □

Combining the previous results, we obtain Theorem 9.

Theorem 9. 2-Disconnected Matching is NP-complete even if the input is restricted to bipartite graphs with diameter 3.

Proof. Let $G = (V_1 \cup V_2, E)$ be a graph generated from the transformation of Section 2.1.2.1. Let us show that this graph is bipartite. Note that the bipartition V_1 of G can be defined by $V_1 = \{l_{iq} \mid i \in [m], q \in \{1,\ldots,6\}\} \cup \{r_{iq} \mid i \in [m], q \in \{7,8,9\}\} \cup \{u_1, u_3 \mid u_1 \in V(U_1), u_3 \in V(U_3)\}$.

Next, we prove that the problem is in NP and NP-hard. Note that a 2-disconnected matching is a certificate to show that the problem is in NP. According to the correspondence between 2-Disconnected Matching and One-in-Three 3SAT solutions described in Lemmas 5 and 4, the One-in-three 3SAT problem, which is NP-complete, can be reduced to 2-Disconnected Matching using a diameter 3 bipartite graph. Therefore, 2-Disconnected Matching is NP-hard and we have proven that 2-Disconnected Matching is NP-complete even for diameter 3 bipartite graphs. □

These results imply the following dichotomies, in terms of diameter.

Corollary 2. For bipartite graphs with diameter $\leq d$, Disconnected Matching is NP-complete if d is at least 3 and belongs to P otherwise.

Corollary 3. For graphs with diameter $\leq d$, Disconnected Matching is NP-complete if d is at least 2 and belongs to P otherwise.

2.1.2.5 Example of 2-Disconnected Matching reduction

Consider the input I with the two clauses $c_1 = (x \vee y \vee z)$ and $c_2 = (w \vee y \vee \overline{x})$.

Let us build the graph G, $V(G) = 30m = 60$, $E(G) = 47m^2 + 24m = 236$, as described in Section 2.1.2.1.

We describe next the edges between B_1 and B_2. Note that in I, the third literal of c_2 is the negation of c_1. Therefore, the edges $\{(r_{11}, l_{23}), (l_{14}, r_{26})\}$ must be added. In addition,

the second literal of c_1 is the same as c_2. Thus, we add the edges $\{(l_{15}, r_{22}), (r_{12}, l_{25}))\}$. The rest of the literals refer to different variables, so there are no additional edges between B_1 and B_2.

We present the only two solutions for One-in-three 3SAT and their corresponding disconnected matchings in G. The assignment of the variables (w, x, y, z), in this order, can be either (T, T, F, F), represented by the matching in Figure 17a, or (F, F, F, T), in Figure 17b. For easier visualization, some edges of B_1 and B_2 are omitted, besides the complete subgraphs H_1 and H_2 and their respective saturated edges. We also show the full graph of the reduction and the corresponding disconnected matching in Figure 18.

2.1.3 NP-completeness for any fixed c

We now generalize our hardness proof to c-Disconnected Matching for every fixed $c \geq 2$.

Lemma 7. For every $c \geq 2$, c-Disconnected Matching is NP-complete on bipartite graphs of diameter 3 and general graphs of diameter 2.

Proof. We proceed by induction on c, where the base case $c = 2$ was handled in Theorem 9. Now, for $c > 2$, let (G, k) be the input to the NP-complete problem $(c-1)$-Disconnected Matching, with G being bipartite and $V(G) = V_1 \dot\cup V_2$ a bipartition. We obtain the input (H, k') c-Disconnected Matching by setting $V(H) = V(G) \cup \{v_1, v_2, w_1, w_2\}$, $E(H) = E(G) \cup \{v_1v_2, w_1w_2, v_1w_2, v_2w_1\} \cup \{aw_2 \mid a \in V_1\} \cup \{bw_1 \mid b \in V_2\}$, and $k' = k+1$. Note that H is bipartite and has diameter 3 since w_i is universal with respect to $V_{3-i} \cup \{v_{3-i}, w_{3-i}\}$. To see that these instances are equivalent, since none of $\{w_1, w_2\}$ can be in a solution M of (H,k'), otherwise $H[M]$ is connected; as such M is an c-disconnected matching if and only if $H[V(G) \cap V(M)] = G[V(M) \setminus \{v_1, v_2\}]$ admits an $(c-1)$-disconnected matching, since any such matching in $H[V(G)]$ can be extended by adding to it the edge v_1v_2.

For the claim about diameter 2 graphs, note that we can identify w_1, w_2 to obtain a diameter 2 graph; the equivalence argument remains unchanged. □

Together with our previous results, Lemma 7 yields a complexity dichotomy for c-Disconnected Matching based on the value of c. We summarize this dichotomy in the following theorem.

Theorem 10. Let $c \geq 1$. The c-Disconnected Matching problem belongs to P if $c = 1$. Otherwise, it is NP-complete even for bipartite graphs of diameter 3 or for general graphs of diameter 2.

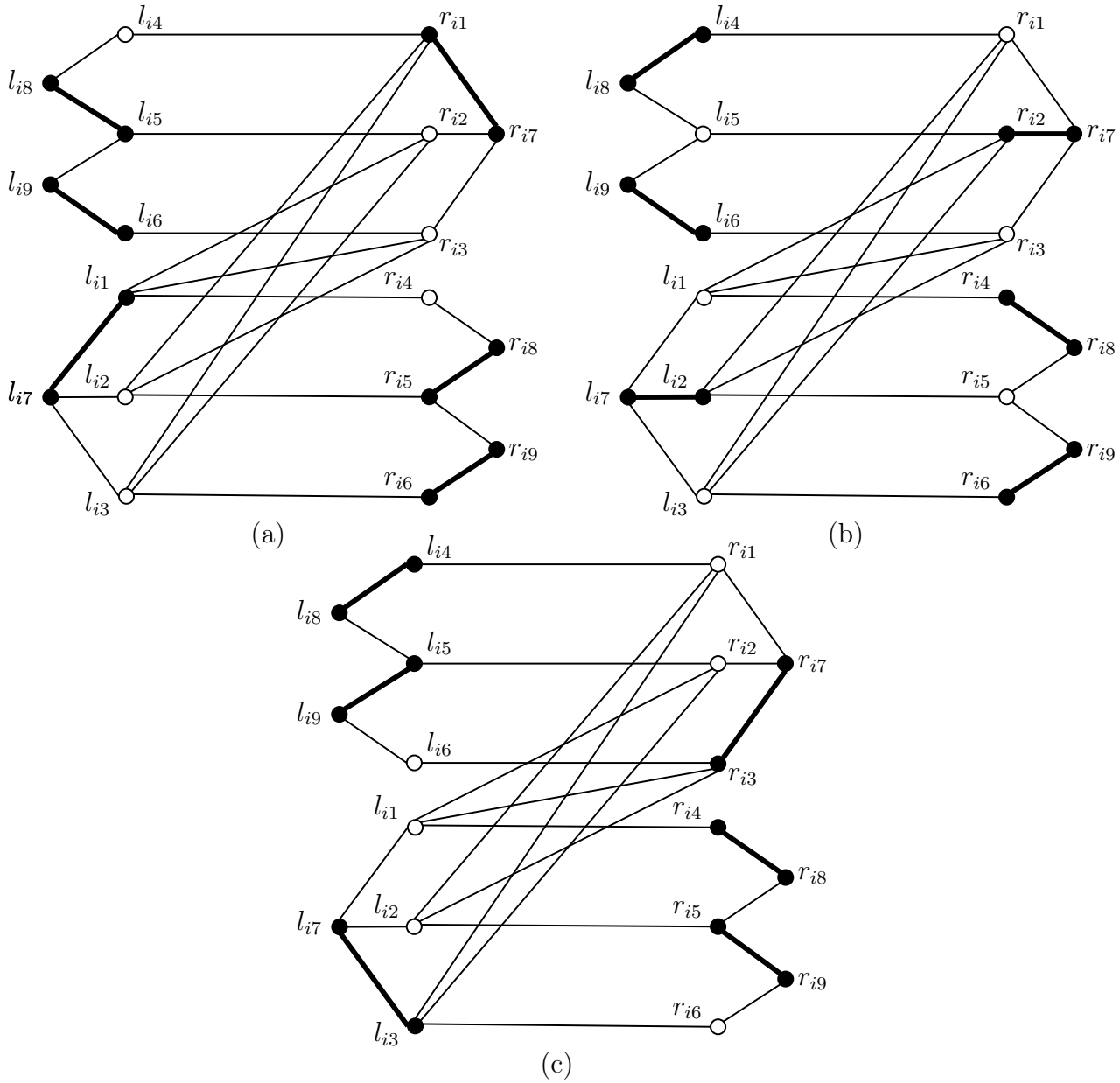


Figure 16 – Subgraph B_i with, in bold, the three sets of vertices possibly saturated with cardinality 12 from Table 6.

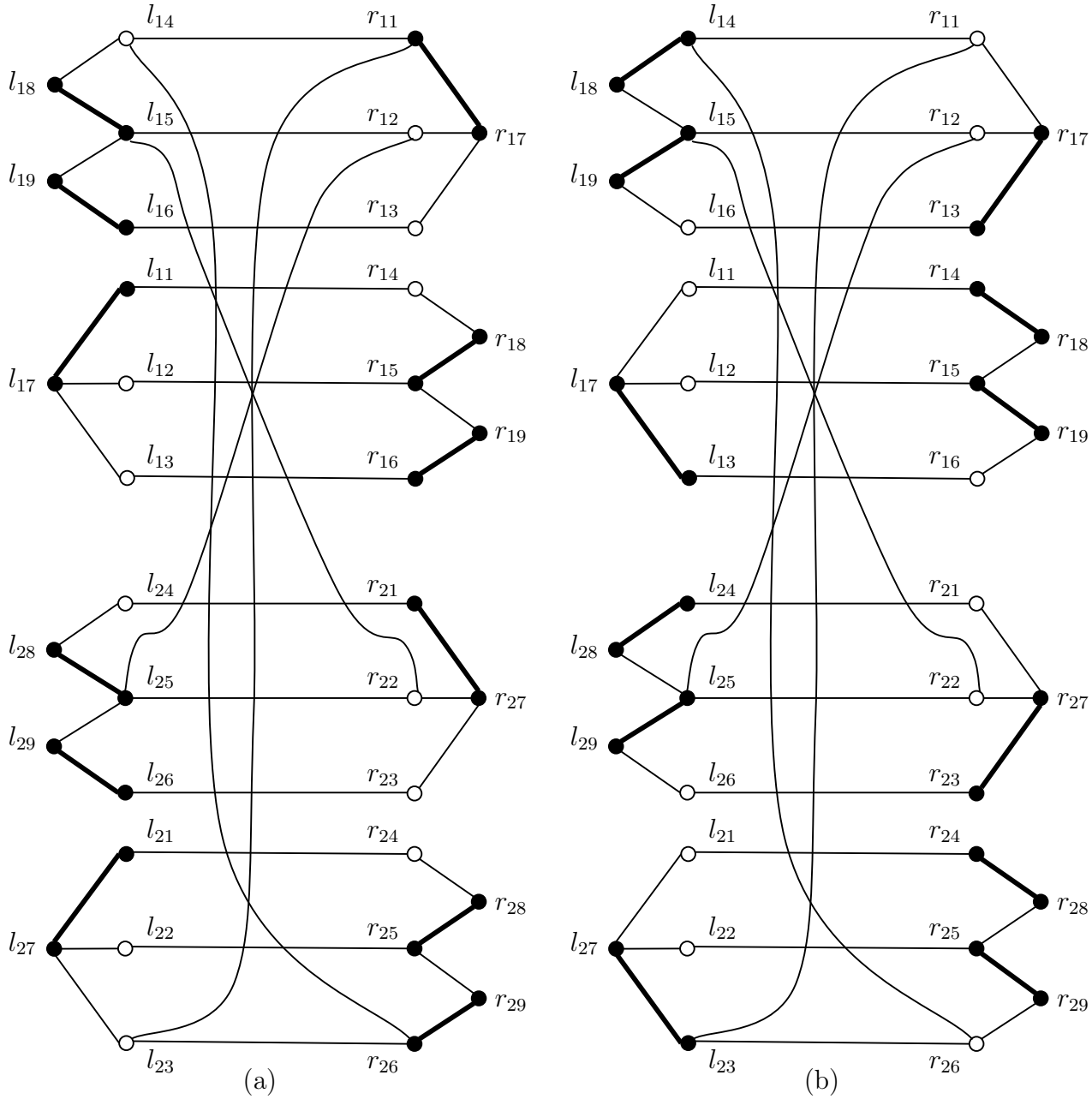


Figure 17 – Disconnected matching examples in the simplified clause subgraphs for the boolean expression $(x \vee y \vee z) \wedge (w \vee y \vee \overline{x})$.

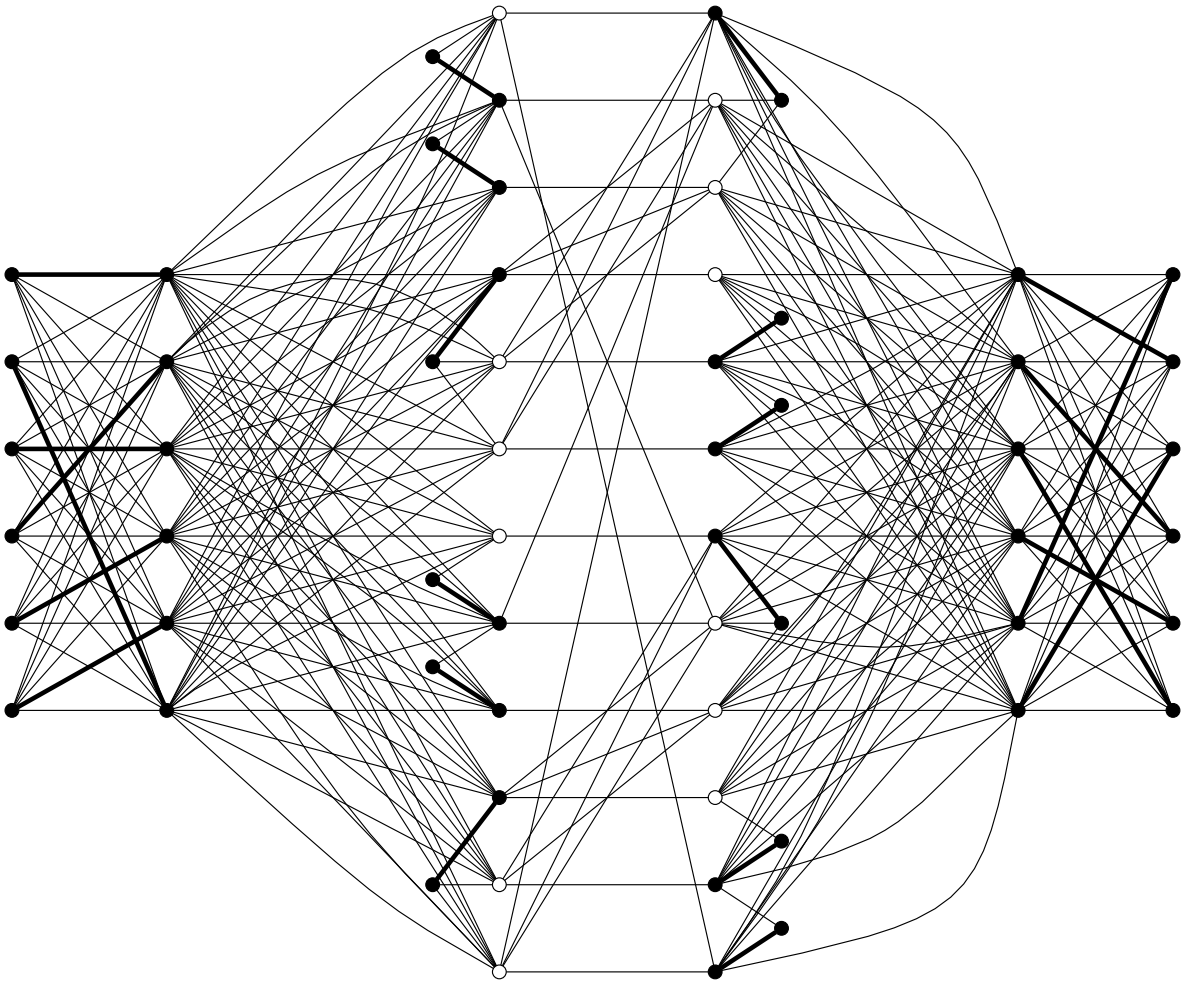


Figure 18 – Example of a full graph generated by our reduction.

2.2 NP-completeness for chordal graphs

In this section, we prove that Disconnected Matching is NP-complete even for chordal graphs with diameter 2. In order to prove it, we describe a reduction from the NP-complete problem Exact Cover By 3-Sets [78]. This problem consists in, given two sets X, $|X| = 3q$, and C, $|C| = m$ of 3-element subsets of X, decide if there exists a subset $C' \subseteq C$ such that every element of X occurs in exactly one member of C'.

For the reduction, we define $c = m - q + 1$, $k = m + 3q$ and build the chordal graph $G = (V,E)$ from the sets C and X as follows.

(I) For each 3-element set $c_i = \{x,y,z\}$, $c_i \in C$, generate a complete subgraph H_i isomorphic to K_5 and label its vertices as $W_i = \{w_{ix}, w_{iy}, w_{iz}, w_i^+, w_i^-\}$.

(II) For each pair of 3-element sets $c_i = \{x,y,z\}$ and $c_j = \{a,b,c\}$ such that $c_i, c_j \in C$, add all edges between vertices of $\{w_{ix}, w_{iy}, w_{iz}\}$ and $\{w_{ja}, w_{jb}, w_{jc}\}$.

(III) For each element $x \in X$, generate a vertex v_x and the edges $v_x w_{ix}$ for every i such that c_i contains the element x.

Note that G is indeed chordal since a perfect elimination order can begin with the simplicial vertices $\{v_1, \ldots, v_{3q}, w_1^+, \ldots, w_m^+, w_1^-, \ldots, w_m^-\}$, and be followed by an arbitrary sequence of the remaining vertices, which induce a clique.

An example of the reduction and its corresponding c-disconnected matching is presented in Figure 19. For better visualization, the edges from rule (II) are omitted.

In Lemmas 8 and 9, we define the polynomial transformation between a $(m - q + 1)$-disconnected matching M, $|M| \geq m + 3q$, and a subset C' that solves the Exact Cover By 3-sets. Then, Theorem 11 concludes the NP-completeness for chordal graphs.

Lemma 8. Given a solution C' for (C,X), a $(m - q + 1)$-disconnected matching M of G with $m + 3q$ edges can be computed in polynomial time.

Proof. Denote the sets of vertices U by $\{w_i^+, w_i^- \mid c_i \in C\}$ and S by the set $\bigcup_{j=1}^{j=m} V(H_j) \backslash U$. Let us build a matching M from the solution C'. For each set $c_i = \{x,y,z\}$ contained in C', add the edges $\{w_{ix}v_x, w_{iy}v_y, w_{iz}v_z\}$ to M. Also, for each set $c_j \in C$, add the edge $w_j^+ w_j^-$.

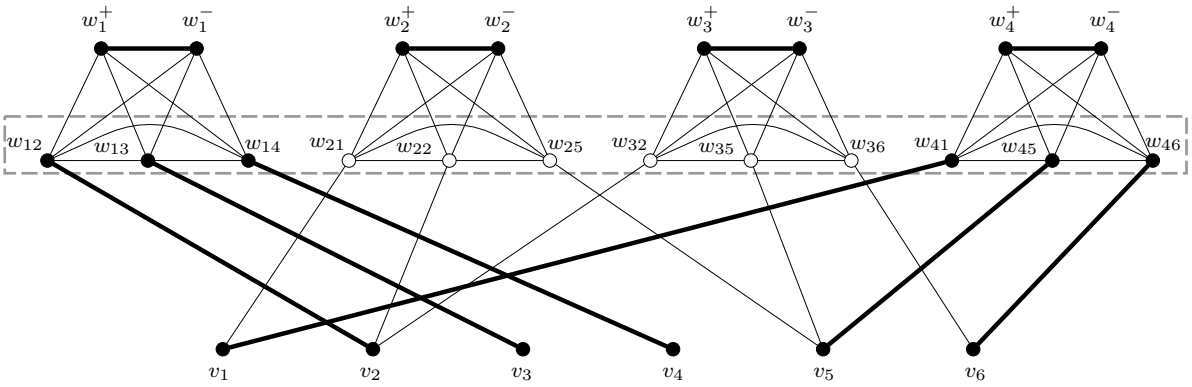


Figure 19 – An example of reduction for the input $X = \{1,2,3,4,5,6\}$ and $C = \{\{2,3,4\}, \{1,2,5\}, \{2,5,6\}, \{1,5,6\}\}$. The subgraph induced by the vertices inside the dotted rectangle is complete and the matching in bold corresponds to the solution $C' = \{\{2,3,4\}, \{1,5,6\}\}$.

Consequently, each H_i such that $c_i \notin C'$ will induce a connected component isomorphic to K_2 in $G[M]$, with the vertices $\{w_i^+, w_i^-\}$, totaling $m-q$ connected components and $2m-2q$ saturated vertices. There will also be one more connected component containing the $3q$ vertices of $\{v_i \mid i \in [3q]\}$ and the $5q$ vertices of $\{W_j \mid c_j \in C \setminus C'\}$. Thus, M saturates $2m - 2q + 3q + 5q = 2m + 6q$ vertices, corresponding to $m + 3q$ edges. Also, $G[M]$ has $m - q + 1$ connected components. So, M is a valid solution for the c-DISCONNECTED MATCHING. □

Lemma 9. Given a $(m - q + 1)$-disconnected matching M of G with $m + 3q$ edges, a solution C' to EXACT COVER BY 3-SETS can be built in polynomial time.

Proof. Denote the sets of vertices U by the set $\{w_i^+, w_i^- \mid c_i \in C\}$ and S by $\bigcup_{j=1}^{j=m} V(H_j) \setminus U$. Consider an arbitrary c-disconnected matching M, $|M| \geq k$ in G. Based on the graph built, we show how such a matching is structured and then build a solution C'. Note that every edge in G is either incident to a vertex of S or to two vertices of U. Since all vertices of S are connected, $G[M]$ can only have two types of connected components:

(I) A K_2 with 2 vertices of U;

(II) A connected component that can contain any vertex, except the ones from type (I) connected components and its adjacencies.

Given that $G[M]$ has at least c connected components, then it must have at least $c-1$ connected components of type (I). For each of them, the 2 saturated vertices are $\{w_i^+, w_i^-\}$, that are contained in U. To keep these vertices in an isolated connected component, the other 3 vertices of $W_i \cap S$ can not be saturated. So, we will not use them for the rest of the construction. In order to have c connected components, it is needed at least one more. Note that, so far, we have $2c - 2 = 2m - 2q$ saturated vertices, and the remaining graph has exactly $6q + 2q = 8q$ vertices. Since M saturates $2k = 2m - 2q$ vertices, then the other connected component, of type (II), must saturate all the remaining $8q$ vertices. Thus, there can be no more than c connected components in $G[M]$ and no more than k edges in M. Note that, for each $i \in [m]$, the subgraph $G[W_i \cap S]$ has either 0 or 3 saturated vertices. If it has 3, each vertex in $W_i \cap S$ must be matched with a vertex of $\{v_j \mid j \in [3q]\}$, by an edge. Such an edge exists because the set c_i has the element j. Therefore, a solution C' can contain the set c_i if and only if $W_i \cap S$ has 3 saturated vertices. □

Theorem 11. DISCONNECTED MATCHING is NP-complete for chordal graphs with diameter 2.

Proof. Note that the c-disconnected matching is a certificate that the problem belongs to NP. We now prove that it is also NP-hard. Lemmas 8 and 9 show that a solution C' for the EXACT COVER BY 3-SETS instance (C,X) corresponds to a c-disconnected matching M of the chordal graph G built by rules (I)-(III), where $c = m - q + 1$ and $|M| = 3q + m$. Note that if we add a universal vertex to G, the same properties hold, and the diameter of G is reduced to 2. For this reason, c-DISCONNECTED MATCHING is NP-hard and, thus, also NP-complete even for chordal graphs. □

We can also make some little modifications to show that DISCONNECTED MATCHING is also NP-complete for bounded vertex degree.

We can also show that the problem is hard even for limited vertex degree graphs. In this case, rule (II) from the previous construction can be replaced by the following.

(II) For each pair of 3-element sets $c_i = \{x,y,z\}$ and $c_j = \{a,b,c\}$ such that $c_i, c_j \in C$, add all edges having one endpoint in $\{w_{ix}, w_{iy}, w_{iz}\}$ and other in $\{w_{ja}, w_{jb}, w_{jc}\}$ if and only if there is an element $p \in c_i$,c_j.

We know that the problem Exact Cover By 3-Sets remains NP-complete even if no element in X appears in more than 3 sets of C [78]. Therefore, G will not be necessarily chordal, though its maximum degree is bounded by a constant.

With the same arguments for the proof given in Theorem 11, we state the following theorem.

Theorem 12. Disconnected Matching is NP-complete even for graphs with maximum degree bounded by a constant and diameter 2.

2.3 Polynomial time algorithms

For our next contributions, we turn our attention to positive results, showing that the problem is efficiently solvable in some graph classes.

2.3.1 Minimal separators and disconnected matchings

It is not surprising that minimal separators play a role when looking for c-disconnected matchings. In fact, for $c = 2$, Goddard et al. [81] showed how to find 2-disconnected matchings in graphs with a polynomial number of minimal separators. We generalize their result by showing that Disconnected Matching parameterized by the number c of connected components is in XP. Note that we do not need to assume that the family of minimal separators is part of the input, as it was shown in [163] that such a family can be constructed in polynomial time.

Theorem 13. Disconnected Matching parameterized by the number of connected components is in XP for graphs with a polynomial number of minimal separators.

Proof. Note that if a matching M is a maximum c-disconnected matching of $G = (V,E)$, then there is a family $\mathcal{S}$ of at most $c - 1$ minimal separators such that $V(G) - V(M)$ contains $\bigcup_{S \in \mathcal{S}} S$. Therefore, if we find such $\mathcal{S}$ that maximizes the size of a maximum matching M in $G[V - (\bigcup_{S \in \mathcal{S}} S)]$ and M is c-disconnected, then M is a maximum c-disconnected matching. Considering that G has $n^{\mathcal{O}(1)}$ many minimal separators, the number of possible candidates for $\mathcal{S}$ is bounded by $n^{\mathcal{O}(c)}$. Computing a maximum matching can be done in polynomial time and checking whether $G[M]$ has c components can be done in linear time. Therefore, the whole procedure takes $n^{\mathcal{O}(c)}$ time and finds a maximum c-disconnected matching. □

In particular, this result implies that c-Disconnected Matching is solvable in polynomial time for chordal graphs [32], circular-arc graphs [43], graphs that do not contain thetas, pyramids, prisms, or turtles as induced subgraphs [3]. We leave as an open question to decide if Disconnected Matching parameterized by c is in FPT for any of these classes.

2.3.2 Interval Graphs

In this section, we show that Disconnected Matching for interval graphs can be solved in polynomial time. To obtain a dynamic programming algorithm, we rely on the ordering property of interval graphs [80]; that is, there is an ordering $\mathcal{Q} = \langle Q_1, \dots Q_p \rangle$ of the p maximal cliques of G such that each vertex of G occurs in consecutive elements of $\mathcal{Q}$ and, moreover the intersection $S_i = Q_i \cap Q_{i-1}$ between two consecutive cliques is a minimal separator of G. Our algorithm builds a table $f(i,j,c')$, where $i,j \in [p]$ and $c' \in [c]$, and is equal to q if and only if the largest c'-disconnected matching of $G\left[\bigcup_{i \leq \ell \leq j} Q_\ell \setminus (S_i \cup S_{j+1})\right]$ has q edges; that is, (G,k,c) is a positive instance if and only if $f(1,p,c) \geq k$.

Theorem 14. Disconnected Matching can be solved in polynomial time on interval graphs.

Proof. We define $|V(G)| = n$ and $|E(G)| = m$. Let (G,k,c) be the input to Disconnected Matching, $\mathcal{Q} = \langle Q_1, \dots, Q_p \rangle$ be an ordering of the maximal cliques of G where each vertex in $V(G)$ occurs in consecutive elements of $\mathcal{Q}$ [80], $\mathcal{S} = \{S_1, S_2, \dots, S_p, S_{p+1}\}$ be a set of minimal separators where $S_i = Q_i \cap Q_{i-1}$, for every $2 \leq i \leq p$, and $S_1 = S_{p+1} = \emptyset$, and $G_{i,j} = G\left[\bigcup_{i \leq \ell \leq j} Q_i \setminus (S_i \cup S_{j+1})\right]$. Our dynamic programming builds a table $f(i,j,c')$, where $i,j \in \{1,\dots,p\}$ and $c' \in \{1,\dots,c\}$, and is equal to q if and only if the largest c'-disconnected matching of $G_{i,j}$ has q edges; that is, (G,k,c) is a positive instance if and only if $f(1,p,c) \geq k$. We define the recurrence relation for f as follows:

$$f(i,j,c') = \begin{cases} -\infty, \text{ if } c' > 1 \text{ and } i = j, \\ \beta(G_{i,j}), \text{ if } c' \in \{0,1\}, \\ \max\limits_{c_1 + c_2 = c'} \max\limits_{i \leq \ell \leq j-1} f(i,\ell,c_1) + f(\ell+1,j,c_2), \text{ otherwise.} \end{cases} \tag{2.1}$$

In the above, $\beta(K_1) = \beta(\emptyset) = -\infty$. If $c' \leq 1$, then, by Theorem 2, $\beta_{d,c}(G_{i,j}) = \beta(G_{i,j}) = f(i,j,c')$ and we are done. Otherwise, let $c \geq 2$, M be a c'-disconnected matching of $G_{i,j}$ with maximum number of edges, C_1, C_2 be two connected components of $G[V(M)]$, and $S \subseteq V(G) \setminus V(M)$ be a non-empty minimal vertex separator between C_1, C_2 in G. Since G is an interval graph, $S \in \mathcal{S}$, say $S = S_\ell$, and w.l.o.g. $C_1 \subset V(G_{i,\ell})$; let c_1 be the number of connected components in $G_{i,\ell}[V(M)]$ and c_2 the same quantity defined in $G_{\ell+1,j}[V(M)]$. By the hypothesis that $S \neq \emptyset$, we have that $|V(G_{i,\ell})|, |V(G_{\ell+1,j})| < |V(G)|$ and, by induction on the size of the input graph, we have that $f(i,\ell,c_1) = \beta_{d,c_1}(G_{i,\ell})$ and $f(\ell+1,j,c_2) = \beta_{d,c_2}(G_{\ell+1,j})$. Finally, since $V(M) \cap S = \emptyset$ and $G \setminus S = G_{i,\ell} \cup G_{\ell+1,J}$, we have that $|M| = f(i,\ell,c_1) + f(\ell+1,j,c_2) \leq f(i,j,c')$ which is covered by the third case of Equation 2.1. For the converse, note that the first two cases of Equation 2.1 are correct: (i) if $i = j$, then $G_{i,j}$ is a clique, and we can have a c'-disconnected matching in $G_{i,j}$ if and only if $c' \leq 1$, and (ii) if $c \leq 1$, then we are asking for a matching with an arbitrary number of connected components, which is what is returned in the second case. For the third case, let ℓ, c_1, c_2 be the values that satisfy $f(i,j,c') = f(i,\ell,c_1) + f(\ell+1,j,c_2)$; M_1 be the corresponding c_1-disconnected matching in $G_{i,\ell}$, M_2 the c_2-disconnected matching in $G_{\ell+1,j}$ and $M' = M_1 \cup M_2$. By induction, $|M_1| = \beta_{d,c_1}(G_{i,\ell})$ and $|M_2| = \beta_{d,c_2}(G_{\ell+1,j})$, M' is a c'-disconnected matching of $G_{i,j}$, and $f(i,j,c') = |M'| \leq |M|$.

As to the running time, since we have $\mathcal{O}(n^2 c)$ entries in the table and each case can be run in $\mathcal{O}(\max\{nc, m\sqrt{n}\})$ time, it holds that the algorithm runs in polynomial time. □

2.3.3 Treewidth

The next result is a standard dynamic programming algorithm on tree decompositions.

Theorem 15. Disconnected Matching can be solved in FPT time when parameterized by treewidth.

Proof. We define $|V(G)| = n$ and $|E(G)| = m$. We suppose w.l.o.g. that we are given a tree decomposition $\mathbb{T} = (T, \mathcal{B})$ of G of width t rooted at an empty forget node; moreover, we solve the more general optimization problem, i.e., given (G,c) we determine the size of the largest c-disconnected matching of G, if one exists, in time FPT on t. As usual, we describe a dynamic programming algorithm that relies on $\mathbb{T}$. For each node $x \in V(T)$, we construct a table $f_x(S, U, \Gamma, \ell)$ which evaluates to ρ if and only if there is a (partial) solution M_x with the following properties: (i) $S \subseteq B_x \cap V(M_x)$ and $G[S]$ admits a perfect matching, (ii) the vertices of $U \subseteq V(M_x) \cap B_x \setminus S$ are half-matched vertices and are going to be matched to vertices in $G \setminus G_x$, (iii) $\Gamma \in \Pi(A_x)$ is a partition of A_x, where $S \cup U \subseteq A_x \subseteq B_x$, and each part of Γ corresponds to a unique connected component of $G[M_x]$, in this case we say that $V(\Gamma) = A_x$, (iv) $G[M_x]$ has *exactly* ℓ connected components that do not intersect B_x, and (v) M_x has ρ edges. If no such solution exists, we define $f_x(S, U, \Gamma, \ell) = -\infty$ and we say the state is *invalid*. Below, we show how to compute each entry for the table for each node type.

Leaf node: Since $B_x = \emptyset$, the only valid entry is $f_x(\emptyset, \emptyset, \{\}, 0)$, which we define to be equal to 0.

Introduce node: Let y be the child of x in T and $B_x = B_y \cup \{v\}$. We compute the table as in Equation 2.2; before proceeding, we define $\Gamma(v)$ to be the block of Γ that contains v. Recall that a partition Γ_1 is a refinement of a partition Γ_2 if, for every block $A_1 \in \Gamma_1$ there is a block $A_2 \in \Gamma_2$ such that $A_1 \subseteq A_2$. In an abuse of notation, a refinement of a block A is a refinement of the partition $\{A\}$. We say that a refinement Γ_2 of a block A is v-connected if for every two blocks $A_1, A_2 \in \Gamma_2$ and every pair of vertices $a_1 \in A_1, a_2 \in A_2$, it holds that $a_1a_2 \notin E(G)$, and every block of Γ_2 has a neighbor of v. Finally, we define the set $\mathsf{ref}_v(\Gamma)$ where $\Gamma' \in \mathsf{ref}_v(\Gamma)$ if and only if: (i) $V(\Gamma') = V(\Gamma) \setminus \{v\}$, (ii) every block in $\Gamma \setminus \Gamma(v)$ is also in Γ', and (iii) Γ' contains a v-connected refinement of $\Gamma(v) \setminus \{v\}$.

$$f_x(S, U, \Gamma, \ell) = \begin{cases} f_y(S, U, \Gamma, \ell), & \text{if } v \notin V(\Gamma); \\ \max\limits_{\Gamma' \in \mathsf{ref}_v(\Gamma)} f_y(S, U \setminus \{v\}, \Gamma', \ell), & \text{if } v \in U; \\ \max\limits_{u \in N(v) \cap S} \max\limits_{\Gamma' \in \mathsf{ref}_v(\Gamma)} f_y(S \setminus \{u,v\}, U \cup \{u\}, \Gamma', \ell), & \text{otherwise.} \end{cases} \tag{2.2}$$

For the first case of the above equation, if $v \notin V(\Gamma)$ then any partial solution M_x of G_x represented by (S, U, Γ, ℓ) is also a solution to G_y under the same constraints since $G_y = G_x \setminus \{v\}$ and v is not in $V(M_x) = V(\Gamma)$. For the second case, let M_x by the solution that corresponds to (S, U, Γ, ℓ), C_v the connected component of $G[V(M_x)]$ that contains v, and $\pi = \{C_1, \ldots, C_q\}$ the (possibly empty) connected components of $G[C_v \setminus \{v\}]$; by definition, $C_v \cap B_x = \Gamma(v)$ is a block of Γ and π is a partition of C_v where vertices in different blocks are non-adjacent. Consequently, we have that $\Gamma' = \Gamma \setminus \Gamma(v) \cup \pi$ is in $\mathsf{ref}_v(\Gamma)$ and $f_y(S, U \setminus \{v\}, \Gamma', \ell)$ is accounted for in the computation of the maximum, which by induction is correctly computed. Finally, for the third case, let $uv \in M_x$, and note that we may proceed as in the previous case: the connected components of $G[C_v \setminus \{v\}]$ induce a v-connected refinement π of $\Gamma(v) \setminus \{v\}$ and $\Gamma' = \Gamma \setminus \Gamma(v) \cup \pi$ is in $\mathsf{ref}_v(\Gamma)$, which results

in a solution of G_y that corresponds to the tuple $(S \setminus \{u,v\}, U \cup \{u\}, \Gamma', \ell)$; since the maximum runs over all neighbors of v in S and over all partitions in ref_v, our table entry is correctly computed.

Forget node: Let y be the child of x in T and $B_x = B_y \setminus \{v\}$. We show how to compute tables for these nodes in Equation 2.4, where $\Gamma^{\{u,v\}} = \Gamma \setminus \{\Gamma(u)\} \cup \{\Gamma(u) \cup \{v\}\}$, $V_{S,U}(\Gamma) = V(\Gamma) \setminus (S \cup U)$, and

$$g(S, U, \Gamma, \ell, u) = \max\{f_y(S, U, \Gamma^{\{u,v\}}, \ell),\ f_y(S \cup \{u,v\}, U, \Gamma^{\{u,v\}}, \ell) + 1\}. \tag{2.3}$$

$$f_x(S, U, \Gamma, \ell) = \max \begin{cases} f_y(S, U, \Gamma, \ell) \\ f_y(S, U, \Gamma \cup \{\{v\}\}, \ell - 1) \\ \max\limits_{u \in V(\Gamma)} f_y(S, U, \Gamma^{\{u,v\}}, \ell), \text{ if } N(v) \cap V(\Gamma) = \emptyset; \\ \max\limits_{u \in N(v) \cap \gamma} g(S, U, \Gamma, \ell, u), \text{ if } \exists \gamma \in \Gamma \mid N(v) \cap V(\Gamma) \subseteq \gamma. \end{cases} \tag{2.4}$$

Let M_x be a solution to G_x represented by (S, U, Γ, ℓ). If $v \notin V(M_x)$, then M_x is a solution to G_y constrained by (S, U, Γ, ℓ) and, by induction, the correctness of f_x is given by the first case of Equation 2.4. Recall that, assuming $v \in V(M_x)$ implies that $N(v) \cap V(\Gamma)$ must be contained in a single block of Γ, otherwise this table entry is deemed invalid and we may safely set it to $-\infty$. If there is some connected component C_v of $G[V(M_x)]$ that has $C_v \cap (V(\Gamma) \cup \{v\}) = \{v\}$, then it must be the case that M_x is a solution of G_y represented by $(S, U, \Gamma \cup \{\{v\}\}, \ell - 1)$ since $C_v \cap B_y = \{v\}$, which is the second case of the equation. Suppose $wv \in M_x$. If $\{u,v\} \subseteq C_v \cap (V(\Gamma) \cup \{v\})$, we branch our analysis on two cases:

1. For the first one, we suppose $N(v) \cap V(\Gamma) = \emptyset$ and note that $\Gamma^{\{u,v\}}(u) = C_v \cap V(\Gamma^{\{u,v\}})$, so we must have that $f_y(S, U, \Gamma^{\{u,v\}}, \ell) \neq -\infty$ is accounted for in the third case of Equation 2.4.

2. Otherwise, there is some $u \in N(v) \cap V(\Gamma)$ and it must be the case that $N(v) \cap V(\Gamma) \subseteq \Gamma(u) = \gamma$. If $u = w$, then M_x is a partial solution to G_y represented by $(S \cup \{u,v\}, U, \Gamma^{\{u,v\}}, \ell)$, so, by induction, $f_y(S \cup \{u,v\}, U, \Gamma^{\{u,v\}}, \ell)$ is well defined, and we have one additional matched edge outside of B_x than outside of B_y, hence the $+1$ term in Equation 2.3. Finally, if $u \neq w$ and $w \notin B_x$, then we proceed as in Case 1, as shown in Equation 2.3, but since v must be in the same connected component of its neighbors, we have fewer entries to check in f_y. Either way, u is accounted for in the range of the maximum in the fourth case of Equation 2.4.

Join node: Finally, let x be a join node with children y,z and $B_x = B_y = B_z$. We obtain the table for these nodes according to the following recurrence relation, where $C = A \sqcup B$ is the *join* between A and B.

$$f_x(S, U, \Gamma, \ell) = \max_{\substack{\ell_y + \ell_z = \ell \\ U_y \dot\cup U_z = V_{S,U}(\Gamma) \\ \Gamma_y \sqcup \Gamma_z = \Gamma}} f_y(S, U \cup U_y, \Gamma_y, \ell_y) + f_z(S, U \cup U_z, \Gamma_y, \ell_z) \tag{2.5}$$

Once again, let M_x be a solution to G_x satisfying (S, U, Γ, ℓ) and $M_i = M_x \cap G_i$ for $i \in \{y,z\}$. Moreover, let $\mathcal{C}_i$ be the connected components of $G[V(M_i)]$, ℓ_i the number of

components in $\mathcal{C}_i$ with no vertex in B_i, and Γ_i the partition of $V(M_i) \cap B_i$ where each block is equal to $C \cap B_i$ for some $C \in \mathcal{C}_i$. Note that it must be the case that $\ell = \ell_y + \ell_z$ — since $M_x = M_y \cup M_z$ — and that $\Gamma = \Gamma_y \sqcup \Gamma_z$ since vertices in different connected components of M_y may be in a same connected component of M_z, but vertices in different connected components in both solutions are not merged in a same connected component of M_x. Now, define $W_y \subseteq V_{S,U}(\Gamma)$ to be the set of vertices that are matched to a vertex of $G_y \setminus S$, let W_z be defined analogously, $U_y = W_z$ and $U_z = W_y$; note that (U_y, U_z) is a partition of $V_{S,U}(\Gamma)$, and that the vertices in $M_i \cap B_i$ that must be matched, but not in G_i, are given by $U \cup U_i$. As such, M_i is represented by $(S, U \cup U_i, \Gamma_i, \ell_i)$ and by induction we have $f_i(S, U \cup U_i, \Gamma_i, \ell_i) = |M_i \setminus S|$, so it holds that $f_x(S, U, \Gamma, \ell) = \sum_{i \in \{y,z\}} f_i(S, U \cup U_i, \Gamma_i, \ell_i) = |M_x \setminus S|$, which is one of the terms of the maximum shown in Equation 2.5.

Recall that we may assume that our tree decomposition is rooted at a forget node r with $B_r = \emptyset$; by definition, our instance (G, k, c) of Disconnected Matching is a YES instance if and only if $f_r(\emptyset, \emptyset, \{\}, c) \geq k$. As to the running time, we have $\mathcal{O}(2^{2t}\eta_t n)$ entries per table of our algorithm, where η_t be the t-th Bell number, each of which can be computed in $\mathcal{O}\big(2^t \eta_{t+1}^2 n\big)$, which is the complexity of computing a join node, so our final running time is of the other of $\mathcal{O}\big(8^t \eta_{t+1}^3 n^2\big)$. □

We observe that Theorem 15 implies tractability for several other parameterizations, including pathwidth and feedback vertex set.

2.4 Kernelization

In the previous section, we presented an FPT algorithm for the treewidth parameterization, which implies tractability for several other parameters, such as vertex cover and max leaf number. In this section, we provide kernelization lower bounds for Disconnected Matching when parameterized by vertex cover and when parameterized by vertex deletion distance to clique. We highlight that these lower bounds hold for the Induced Matching problem and, for the former parameterization, even when restricted to bipartite graphs.

The proofs are through OR-cross-compositions from the Exact Cover by 3-Sets problem, and are inspired by the proof of Section 2.2. Throughout this section, let $\mathcal{S} = \{(X_1, C_1), \ldots, (X_t, C_t)\}$ be the input instances to Exact Cover by 3-Sets; w.l.o.g., we assume that $X_i = X = [n]$ and $|C_i| = m$ for all $i \in [t]$, and define $\mathcal{C} = \bigcup_{i \in [t]} C_i$. We further assume that, for any two instances, $C_i \neq C_j$, which implies that $C_i \setminus C_j$ and $C_j \setminus C_i$ are non-empty. We denote by (G,k) the built Induced Matching instance.

2.4.1 Vertex Cover

Construction. We begin by adding to G the set $A = \{v_a \mid a \in X\}$ and, for each set $S_j \in \mathcal{C}$ where $S_j = \{a,b,c\}$, we add one copy Q_j of $K_{1,4}$, with vertices labeled as $\{w_j, w_j^*, w_{ja}, w_{jb}, w_{jc}\}$; w_j is the central vertex, while w_{ja}, w_{jb}, w_{jc} are the *interface* vertices of Q_j. Then, we add edges to G so $w_{ja}v_d \in E(G)$ if and only if $a = d$. Now, we add to G an instance selector gadget I, which is simply a star with t leaves, with the central vertex labeled as q and the i-th leaf labeled as p_i. To complete the construction of G, for each $p_i \in V(I)$ and $S_j \in \mathcal{C} \setminus C_i$, we add all edges between p_i and the interface vertices of Q_j, i.e. if S_j is not a set of the i-th instance, we add edges between Q_j and p_i. Finally, we set $k = n + |\mathcal{C}| - \frac{n}{3} + 1$.

Lemma 10. Graph G is bipartite and has a vertex cover of size $\mathcal{O}(n^3)$.

Proof. We construct the bipartition (Y,W) as follows: $Y = (I \setminus \{q\}) \cup \{w_j \mid S_j \in \mathcal{C}\} \cup A$ and $W = V(G) \setminus Y$. To see that Y is an independent set, note that: (i) each of its three components induces an independent set of G, (ii) $I \setminus \{q\}$ is not adjacent to the central vertex of any Q_j nor to any vertex of A, and (iii) vertices of A are non-adjacent to the central vertices of the Q_j's. For W, note it is composed by the leaves of the Q_j's, which together form an independent set, and vertex q, which is only adjacent to vertices of I, none of which belong to W. Note $D = V(G) \setminus (V(I) \setminus \{q\})$ is a vertex cover, i.e. $G \setminus D$ is an independent set, since each connected component of $G \setminus D$ corresponds to a leaf of I. Observe that $|D| = 5|\mathcal{C}| + n + 1$ and that there are most $\binom{n}{3}$ elements in $\mathcal{C}$ since there are at most this many subsets of three distinct elements of the ground set X, so $|D| = \mathcal{O}(n^3)$. □

Lemma 11. If an Exact Cover by 3-sets instance $(X_\ell, C_\ell) \in \mathcal{S}$ admits a solution, then G admits an induced matching with k edges.

Proof. Let Π be the solution to (X_ℓ, C_ℓ), $S_j \in \mathcal{C}$, and $S_j = \{a,b,c\}$. We add to M the edges $\{w_{ja}v_a, w_{jb}v_b, w_{jc}v_c\}$, if $S_j \in \Pi$, otherwise we add edge $w_jw_j^*$ to M, totalling $n + |\mathcal{C}| - \frac{n}{3}$ edges. For the final edge, add qp_ℓ to M. In terms of connected components, each edge of M is a distinct component, since: (i) each Q_j either has 3 of its leaves in M but not its central vertex, or it has its central vertex in M, (ii) each vertex of A is adjacent to only one saturated vertex, i.e. its unique neighbor in $V(M)$ is w_{ja}, and (iii) p_ℓ is adjacent only to interface vertices that are not saturated by M, so its unique neighbor in $V(M)$ is q. As such, M is an induced matching with k edges. □

Let us now show the converse.

Lemma 12. In every induced matching M of size k of G, q is M-saturated.

Proof. Towards a contradiction, suppose that $q \notin V(M)$ and, furthermore that $I \cap V(M) = \emptyset$. In this case, note that $|M| \leq n + |\mathcal{C}| - \frac{n}{3} = k - 1$, since we may have at most $\frac{n}{3}$ stars Q_j with the three interface vertices in M, contributing with n edges to M, and all other Q_j's have at most edge $w_jw_j^*$ in M, totaling $n + |\mathcal{C}| - \frac{n}{3}$ edges in the matching.

If, on the other hand, $I \cap V(M) \neq \emptyset$, then suppose $p_i \in V(M)$. Since $q \notin V(M)$, p_i is matched with a vertex in Q_j, say w_{ja}, which implies that $Q_j \cap V(M) = \{w_{ja}\}$, since p_i is adjacent to all three interface vertices of Q_j and, if w_j^* is saturated by M, then w_j also is, which is impossible since M is an induced matching. Moreover, note that, for every Q_x with vertices adjacent to p_i, we have that $E(Q_x) \cap M \subseteq \{w_xw_x^*\}$. At this point, we have accounted for $1 + (|\mathcal{C}| - m - 1) = |\mathcal{C}| - m$ edges of M. For the m Q_j's with no vertex adjacent to p_i, they can each contribute with at most three edges to M but no more than $n + m - \frac{n}{3}$ in total, since each Q_j will either: (i) have some of its interface vertices matched to vertices $\{v_a, v_b, v_c\}$, (ii) have w_j^* saturated, or (iii) have exactly one of its interface vertices saturated to either some other p_y or to w_j. As such, we have at most n edges coming from the first option, while the others amount to, at most $m - \frac{n}{3}$ additional edges. Finally, this implies that $|M| \leq |\mathcal{C}| - m + n + m - \frac{n}{3} = n + |\mathcal{C}| - \frac{n}{3} < k$, and we conclude that q must be M-saturated. □

As a consequence of the previous lemma, there is an edge of the form qp_i in every solution to (G,k).

Lemma 13. If (G,k) admits a large enough induced matching, then at least one instance $(X_\ell, C_\ell) \in \mathcal{S}$ also admits a solution.

Proof. Let M be a solution to (G,k) with $qp_\ell \in M$ and $\mathcal{Q}$ be the set of Q_j's with at least one saturated interface vertex. Note that no vertex in $Q_j \in \mathcal{Q}$ is adjacent to p_ℓ, otherwise $V(M)$ would not induce a matching. Let us show that $|\mathcal{Q}| = \frac{n}{3}$. If we had any more elements in $\mathcal{Q}$, M would have at most n edges incident to an interface vertex and at most $|\mathcal{C}| - |\mathcal{Q}|$ edges incident to the non-interface vertices of Q_j's, which implies that $|M| \leq 1 + n + |\mathcal{C}| - |\mathcal{Q}| < 1 + n + |\mathcal{C}| - \frac{n}{3} = k$; the first property follows from the fact that q is already in $V(M)$ and the neighbors of interface vertices, aside outside of A and the central vertex of Q_j, are also neighbors of q. On the other hand, if $|\mathcal{Q}| < \frac{n}{3}$, then we would have that $|M| \leq 1 + 3|\mathcal{Q}| + |\mathcal{C}| - |\mathcal{Q}| < 1 + |\mathcal{C}| + 2\frac{n}{3} = k$. With this in hand, note that, to obtain $|M| = k$, it must be the case that $|V(M) \cap \bigcup_{Q_j \in \mathcal{Q}} Q_j| = n$ and interface vertices are matched with vertices of A. This holds since:

$$\begin{aligned} k \geq |M| &= |M \cap E(I)| + \sum_{S_j \in \mathcal{C}} |M \cap E(Q_j)| \leq 1 + 3|\mathcal{Q}| + |\mathcal{C}| - |\mathcal{Q}| \\ &= 1 + n + |\mathcal{C}| - \frac{n}{3} = k \end{aligned}$$

Moreover, the elements of $\mathcal{Q}$ must not be adjacent to p_ℓ, which implies that, for each $Q_j \in \mathcal{Q}$, we have that $S_j \in C_\ell$. Since vertices $\{a,b,c\}$ of A are matched with vertices $\{w_{ja}, w_{jb}, w_{jc}\}$ of $Q_j \in \mathcal{Q}$, it follows that $S_j = \{a,b,c\}$ and that $\{S_j \mid Q_j \in \mathcal{Q}\}$ is a solution to (X_ℓ, C_ℓ). □

Finally, combining Lemmas 10, 11, and 13, we obtain the kernelization lower bound.

Theorem 16. Induced Matching does not admit a polynomial kernel when jointly parameterized by vertex cover and solution size unless NP ⊆ coNP/poly, even when restricted to bipartite graphs.

Corollary 4. Disconnected matching does not admit a polynomial kernel when jointly parameterized by vertex cover and number of edges in the matching unless NP ⊆ coNP/poly, even when restricted to bipartite graphs.

2.4.2 Distance to Clique

It is worth noting at this point that the proof we have just presented can be adapted to the distance to clique parameterization without significant changes. To do so, we replace I with a clique of size $t+1$, label its vertices arbitrarily as $\{q, p_1, \ldots, p_t\}$ and proceed exactly as before. The caveat is that we must show that any solution that picks an edge $p_x p_y$ can be changed into a solution that picks, say, qp_x and that this new solution behaves in the exact same way as the one we outline in Lemma 13. We prove this in the following lemma.

Lemma 14. If (G,k) admits a solution, then it also admits a solution where q is saturated.

Proof. Let M be a solution to (G,k) that does not saturate q. Our first claim is that $M \cap E(I) \neq \emptyset$. Note that no p_i may be matched to a vertex outside of I; we could immediately repeat the second paragraph of the proof of Lemma 12, so if $M \cap E(I) \neq \emptyset$,

it holds that $V(M) \cap I = \emptyset$. Now, observe that $|M| \leq k-1$ since the maximum induced matching in $G \setminus I$ uses as many edges between A and the Q_j's as possible and, for each Q_j without M-saturated interface vertices, we pick edge $w_j w_j^*$, totalling at most $|A| + |\mathcal{C}| - \frac{n}{3} \leq k-1$ edges in M. As such, $p_x p_y \in M$ these are the only vertices saturated in I, otherwise M would not be induced. Replacing edge $p_x p_y$ by edge $q p_y$ maintains the property that M is an induced matching and does not change its cardinality, completing the proof. □

At this point, we can immediately repeat the proof of Lemma 13. Together with Lemma 14, we observe that (G,k) admits a solution if and only if some instance (X, C_i) also does. Finally, by observing that the same vertex cover described in Lemma 10 is a clique modulator for the current construction, we obtain the following theorem.

Theorem 17. Induced Matching does not admit a polynomial kernel when jointly parameterized by vertex deletion distance to clique and solution size unless NP ⊆ coNP/poly.

Corollary 5. Disconnected matching does not admit a polynomial kernel when jointly parameterized by vertex deletion distance to clique and number of edges in the matching unless NP ⊆ coNP/poly.

2.5 Equality of disconnected matching number and matching number

In this section, we prove that, given a graph G, deciding if $\beta_{d,i}(G) = \beta_{d,j}(G)$ is NP-hard for every fixed i and j, $1 \leq i < j \leq \beta_*$, even for diameter 3 bipartite graphs. For the particular case where $i = 1$, the equality is equivalent to $\beta(G) = \beta_{d,j}(G)$, whose decision is NP-complete, since it is also in NP.

Note that, depending on the graph, the disconnected matching and the matching numbers, β_d and β respectively, can be the same. For instance, in a P_6, $3 = \beta(P_6) > \beta_d(P_6) = 2$, while, in a P_5, $2 = \beta(P_5) = \beta_d(P_5)$.

In order to prove the NP-hardness result, we first describe a reduction to the decision $\beta(G) = \beta_d(G)$ from the NP-complete problem Exact Cover By 3-Sets [78], whose input is $(\mathcal{C},X)$. Now, we build the input bipartite graph $G = (V_1 \dot\cup V_2, E)$ as follows.

(I) Generate a complete bipartite subgraph, $H = (V_H \dot\cup V'_H, E_H)$, isomorphic to $K_{q,|\mathcal{C}|}$. Let V_H be the bipartition of size $|\mathcal{C}|$ from H, whose vertices are labeled as $\{h_i \mid C_i \in \mathcal{C}\}$.

(II) For each $C_i \in \mathcal{C}$, generate a copy of P_3 whose endpoints are labeled u_i^+ and u_i^-. Connect the other vertex, u_i, to h_i.

(III) For each $x_i \in X$, generate the subgraph Y_i, isomorphic to K_{f_i, f_i-1} where f_i is the number of triples in $\mathcal{C}$ that contains the element x_i. Moreover, label the vertices of the bipartition of size f_i as $\{w_{i,j} \mid x_i \in C_j \in \mathcal{C}\}$. Add the edges $\{u_i w_{i,j} \mid x_i \in C_j \in \mathcal{C}\}$. Let W_i^+ and W_i^- be respectively the set of vertices of bipartitions of size f_i and $f_i - 1$.

(IV) Add two copies of K_2, whose vertices are labeled $\{t^+,t^-\}$ and $\{b^+,b^-\}$. Connect b^- to $W_i^+ \cup \{u_i^- \mid C_i \in \mathcal{C}\}$ and t^- to $V_H \cup \{u_i^+ \mid C_i \in \mathcal{C}\}$.

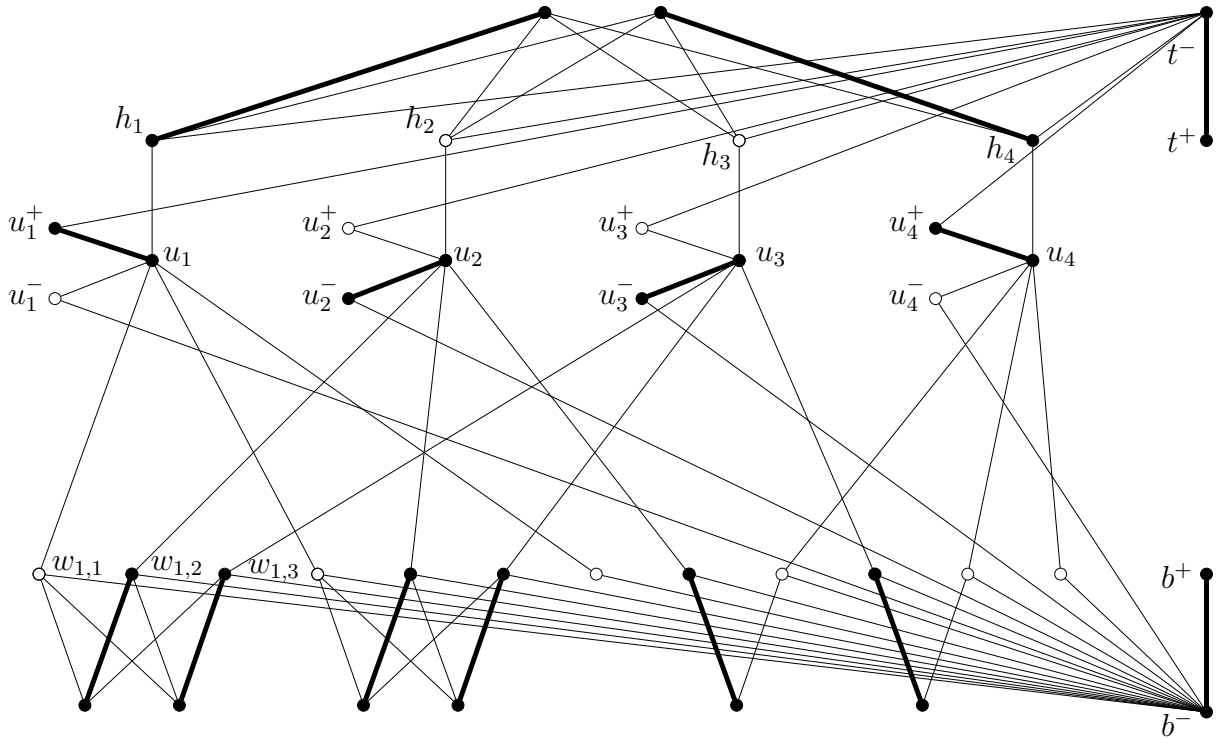


Figure 20 – Example of a bipartite graph built from the input of triples $\{\{x_1,x_2,x_3\}, \{x_1,x_2,x_4\}, \{x_1,x_2,x_5\}, \{x_4,x_5,x_6\}\}$ of Exact Cover By 3-Sets . The matching corresponds to the solution $\{\{x_1,x_2,x_3\},\{x_4,x_5,x_6\}\}$.

Note that G is indeed bipartite, as we define its bipartitions as $V_1 = \{v \mid x_i \in X, v \in W_i^-)\} \cup \{u_i \mid C_i \in \mathcal{C}\} \cup V'_H \cup \{b^-,t^-\}$, and $V_2 = \{N(u_i) \mid C_i \in \mathcal{C}\} \cup \{t^+,b^+\}$.

Observe in Figure 20 an example of this reduction graph for the input $X = \{x_1, \ldots, x_6\}$ and $\mathcal{C} = \{\{x_1,x_2,x_3\}, \{x_1,x_2,x_4\}, \{x_1,x_2,x_5\}, \{x_4,x_5,x_6\}\}$, in addition to a maximum matching corresponding to solution $\mathcal{C}' = \{\{x_1,x_2,x_3\}, \{x_4,x_5,x_6\}\}$.

Lemma 15. A maximum matching of G has cardinality $4|\mathcal{C}| - 2q + 2$ and saturates all vertices of V_1.

Proof. We define a matching M of size $4|\mathcal{C}| - 2q + 2$ that saturates all vertices of the bipartition V_1, which is strictly smaller than V_2. Let us define M by the union of four edge sets, $S_1 = \{b^+b^-, t^+t^-\}$, $S_2 = \{u_iu_i^+ \mid C_i \in \mathcal{C}\}$, S_3 as q disjoint edges of H, and S_4 as, for each $x_i \in X$, $f_i - 1$ disjoint edges of Y_i. Observe that $|S_1| = 2$, $|S_2| = |\mathcal{C}|$, $|S_3| = q$, and $|S_4| = 3\mathcal{C} - |X| = 3\mathcal{C} - 3q$. So, M is indeed maximum and has cardinality $4|\mathcal{C}| - 2q + 2$. □

Lemma 16. The answer to the input $(X,\mathcal{C})$ to Exact Cover By 3-sets is YES if and only if there is a disconnected matching M with cardinality $4|\mathcal{C}| - 2q + 2$ in G. Moreover, if M exists, $G[M]$ has exactly two connected components.

Proof. ($\Rightarrow$): Let $\mathcal{C}' \subseteq \mathcal{C}$ be a solution to $(X,\mathcal{C})$ of Exact Cover By 3-sets. We saturate the set of vertices $\{h_i \mid C_i \in \mathcal{C}'\}$ with any set of edges from H. Also, we saturate $\{b^+b^-, t^+t^-\}$, and, for each $x_i \in X$, all vertices of Y_i except $w_{i,j}$, $C_j \in \mathcal{C}'$, by any set of edges of Y_i. This is possible because $Y_i - w_{i,j}$ is a complete bipartite subgraph. Moreover, for each $C_i \in \mathcal{C}$, saturate $u_iu_i^+$ if $C_i \in \mathcal{C}'$ or $u_iu_i^-$ otherwise. Note that $G[M]$ has exactly

two connected components, one containing $\{t^-,t^+\} \cup \{h_i,u_i,u_i^+ \mid C_i \in \mathcal{C}'\}$ and, other, $\{b^-,b^+\} \cup \{u_i,u_i^- \mid C_i \in (\mathcal{C} \setminus \mathcal{C}')\} \cup \{V(Y_i) \setminus \{w_{i,j}\} \mid x_i \in X, C_j \in \mathcal{C}'\}$. The corresponding separator that disconnects such components is $\{u_i^+ \mid C_i \in (\mathcal{C} \setminus \mathcal{C}')\} \cup \{u_i^- \mid C_i \in \mathcal{C}'\} \cup \{w_{i,j} \mid x_i \in X, C_j \in \mathcal{C}'\}$.

($\Leftarrow$): Let M be a maximum disconnected matching with the same size of a maximum matching. By Lemma 2.5, we know that $|M| = 4|\mathcal{C}| - 2q + 2$ and M saturates all vertices of V_1. Note that t^- and b^- since they are contained in V_1. Initially, we replace the saturated edges of M incident to t^- and b^- by t^+t^- and b^+b^-, in the case those edges were not in M. A solution for Exact Cover By 3-sets can be defined as $\mathcal{C}' = \{C_i \mid C_i \in \mathcal{C}, u_i^+ \in V(M)\}$. We prove the correctness in the following.

Observe that every edge of G is adjacent to either t^-t^+ or b^-b^+, which implies $G[M]$ has exactly two connected components, each containing one of these edges.

Note that, for every $C_i \in \mathcal{C}$, the vertex u_i is saturated, as it is in V_1. Also, for each $x_i \in X$, as Y_i is a complete bipartite subgraph whose bipartitions sizes differ by 1, clearly there is at most one vertex not saturated by M in W_i^+. So, there are at most $3q$ vertices not saturated in $\{W_i^+ \mid x_i \in X\}$.

Also, if the vertex h_i is saturated, then u_i, which is also saturated, is in the same connected component of t^-. For this reason, the three adjacent vertices of $\{W_i^+ \mid x_i \in X\} \cap N(u_i)$ are not saturated; otherwise M would not be disconnected as it would connect the component of t^- with the one of b^-. In this case, $u_iu_i^+ \in M$, as u_i^+ is the only neighbor of u_i left that can be saturated.

There are at least q vertices saturated in V_H, since H is a complete bipartite subgraph and $q = |V'_H| \geq |V_H|$. For this reason, there are at least $3q$ vertices not saturated in $\{W_i^+ \mid x_i \in X\}$. As we stated previously, there are also at most $3q$ vertices not saturated in this set. So, there is exactly one vertex not saturated in each W_i^+, $x_i \in X$. Observe that those vertices must be adjacent to vertices u_i such that $\{h_i,u_i,u_i^+\}$ are saturated. If h_i is not saturated, then $u_iu_i^- \in M$, since all vertices of $\{W_i^+ \mid x_i \in X\} \cap N(u_i)$ are saturated and part of the connected component of b^-. Therefore, $u_iu_i^+ \in M$ if h_i is saturated or $u_iu_i^- \in M$ otherwise.

Hence, the q saturated vertices of type u_i are disjointly adjacent to non saturated vertices of distinct Y_j subgraphs, which represent elements $x_j \in C_i$. Therefore, $\mathcal{C}'$ is a solution to (X,C) of Exact Cover By 3-sets. □

Lemma 16 is enough to prove the NP-completeness of deciding if $\beta(G) = \beta_d(G)$. Next, we go further, showing the NP-hardness of $\beta_{d,i}(G) = \beta_{d,j}(G)$, for fixed arbitrary i and j such that $1 \leq i < j \leq \beta_*$. Besides, we also strengthen our results in terms of diameter of the input graph. For this purpose, we add the following rules to the construction of the reduction graph G.

(V) Generate $j - 2$ copies of K_2.

(VI) Generate one copy of K_2 and connect each of its vertices to all vertices of each bipartition of G.

Theorem 18. Given a graph G, deciding if $\beta_{d,i}(G) = \beta_{d,j}(G)$ is NP-hard even for diameter three bipartite graphs for every fixed i and j, $1 \leq i < j \leq \beta_*$.

Proof. Observe that vertices from (VI) are never part of a maximum disconnected matching since any edge adjacent to it would connect all the matching. Also, edges from

(V), which we denote by M'', are always part of a maximum c-disconnected matching, for any c.

Let G' be the subgraph of G defined by rules (I), (II), (III) and (IV) and M'_i be a maximum matching in G'. Note that a maximum i-disconnected matching in G can be defined as $M_i = M'_i \cup M''$. This holds because $G[M_i]$ has at least $j - 1 \geq i$ connected components, $j - 2$ induced from M'' and at least one from M'_i. In order to exist a j-disconnected matching, then G' must admit a maximum matching M'_j, such that $|M'_j| = |M'_i|$ and $G[M'_j]$ has two connected components. By Lemma 16, this is true if and only if $(X,\mathcal{C})$ has answer YES to X3C. If there is such matching M'_j, then $M'' \cup M'_j$ is a maximum j-disconnected matching. Otherwise, there is no j-disconnected matching having cardinality $|M_i|$. Therefore, deciding if $\beta_{d,i}(G) = \beta_{d,j}(G)$ is NP-hard. □

For the case where $i = 1$, deciding if $\beta(G) = \beta_{d,j}(G)$ is clearly in NP, as $\beta(G)$ can be calculated in polynomial time, and a certificate to it can be a j-disconnected matching of the same size. Thereby, we enunciate the following corollary, considering the case where $i = 1$ and $j = 2$.

Corollary 6. Given a graph G, deciding if $\beta(G) = \beta_d(G)$ is NP-complete even for diameter three bipartite graphs.

3 CONNECTED MATCHINGS

In this chapter, we study the $\mathscr{P}$-matching for the property of being connected, presenting a linear algorithm to obtain it when given a maximum matching. Also, we study its weighted version, which is NP-complete for the general case, in various graph classes and weight constraints.

A connected matching M is a matching such that $G[M]$ is connected. We approach its optimization problem, Maximum Connected Matching.

Maximum Connected Matching
Instance: A graph G.
Task: Find a connected matching M of G with maximum cardinality.

We also work with the weighted versions of connected matchings, by both decision and optimization problems, denoted respectively by Weighted Connected Matching and Maximum Weight Connected Matching.

Weighted Connected Matching
Instance: An edge-weighted graph G and an integer k.
Question: Is there a connected matching M of weight at least k?

Maximum Weight Connected Matching
Instance: An edge-weighted graph G.
Task: Find a connected matching M of G of maximum weight.

The rest of this chapter is organized as follows. In Section 3.1, we give an alternative proof to the fact that Maximum Connected Matching can be solved in polynomial time, presenting an algorithm for it.

Then we turn our attention to the weighted variant of the connected matchings problem, Weighted Connected Matching, under some different constraints. Our investigation begins with starlike graphs, a subclass of chordal graphs, where we show that the problem is NP-complete when arbitrary weights are allowed. On the other hand, we present a polynomial-time algorithm for chordal graphs if all weights are non-negative.

Afterwards, we turn our attention to bipartite graphs, first showing that even for binary weights, Weighted Connected Matching remains NP-complete, even if the graph is planar or its diameter is bounded. If negative weights are allowed, we prove that the problem remains hard for subcubic planar graphs.

We then show some positive cases, namely (i) graphs of maximum degree two, establishing a complexity dichotomy based on the maximum degree, (ii) graphs of bounded treewidth, and (iii) trees; the latter is an improvement upon the treewidth algorithm since it runs in linear time. The treewidth algorithm implies fixed-parameter tractability for the treewidth parameterization, which leads us to the final result, that no polynomial kernel exists when parameterizing by vertex cover unless NP $\subseteq$ coNP/poly. The main results are summarized in Table 3.

Table 3: Summary of our results for Weighted Connected Matching.

<table>
<tr><td colspan="2" rowspan="2">Graph class</td><td colspan="2">Complexity</td></tr>
<tr><td>Weights ≥ 0</td><td>Any weights</td></tr>
<tr><td colspan="2">General</td><td colspan="2" rowspan="2">NP-complete
(Theorem 23)</td></tr>
<tr><td colspan="2">Bipartite having diameter at most 4</td></tr>
<tr><td colspan="2">Chordal</td><td rowspan="2">P
(Theorem 22)</td><td rowspan="2">NP-complete
(Theorem 21)</td></tr>
<tr><td colspan="2">Starlike</td></tr>
<tr><td rowspan="2">Planar</td><td>bipartite</td><td colspan="2">NP-complete
(Theorem 24)</td></tr>
<tr><td>subcubic</td><td>?</td><td>NP-complete
(Theorem 25)</td></tr>
<tr><td colspan="2">$\Delta \leq 2$</td><td colspan="2">P
(Theorem 26)</td></tr>
<tr><td colspan="2">Tree</td><td colspan="2">P
(Theorem 29)</td></tr>
</table>

3.1 Linear algorithm for connected matchings in general graphs

Our result for Maximum Connected Matching is a linear algorithm, which is based on the proof that $\beta(G) = \beta_c(G)$, i.e. a maximum connected matching is also a maximum matching, presented in Theorem 2 of [81]. In this Theorem, it is presented a procedure that turns a maximum matching into a maximum connected matching. Nevertheless, they do not give evidence or signs that it can be done in linear time, the complexity that we reach in Theorem 19. In the following, we also describe this result in the form of an algorithm.

Theorem 19. Given a maximum matching in an arbitrary graph G, a maximum connected matching in G can be found in linear time.

Proof. Let M be a maximum matching of the input graph, and $r \in V(M)$. We begin by running a BFS search on $G[M]$, starting at r, in order to obtain the connected component S of $G[M]$ that contains r.

Note that every vertex in $N(S)$ is not saturated, since S is maximal. Also, by the maximality of M, for each non-saturated vertex v it holds that $N(v)$ is saturated. Hence, we repeat the following operation: while there is a vertex $v \in N(S)$ having any neighbor $w \notin S$, replace the edge saturating w by vw. Note that this operation increases the size of S by at least two vertices, namely v and w, but does not change the number of vertices in the matching.

This can be implemented in linear time since we never need to look at any neighborhood of a vertex of S twice; moreover, whenever a vertex $v \notin N(S)$ is considered, either v is included in S or its neighborhood is a subset of S and v will never be a candidate to be added to S. □

Note that, to use the algorithm described in Theorem 19, it is necessary to have calculated a maximum matching before. To obtain a maximum matching, an algorithm with complexity $\mathcal{O}(m\sqrt{n})$ is known [138]. Therefore, this complexity is the same as the Maximum Connected Matching.

Corollary 7. MAXIMUM CONNECTED MATCHING has the same time complexity of MAXIMUM MATCHING.

For some classes, such as trees, block graphs, and chordal bipartite graphs, MAXIMUM MATCHING can be solved with better time complexities. Hence, for these classes, the algorithm we describe shows that MAXIMUM CONNECTED MATCHING also can be solved with the same complexity.

As a complement to Theorem 19, we show an algorithm to obtain a maximum connected matching, given a maximum matching. It uses two queues, Q_s and Q_n, to store saturated vertices and non-saturated vertices by M, respectively. Besides, we use a set C, to which vertices of C_r or new vertices are added. The main loop adds at least one vertex to C in each iteration, and it contains two other auxiliary loops. The first auxiliary loop calls EXPAND(v) for each vertex v of Q_s, analyze $N(v)$, and appropriately add to the queue each vertex in the corresponding neighborhood that has not been added to it yet. The second auxiliary loop calls function TRYTOCONNECT(v) for each vertex v of Q_n. If there is a vertex $w \in N(v) \setminus C$, then w is incident to a matched edge, say wu. So, we exchange edges in M, removing wu and adding vw.

Algorithm 1 Maximum Connected Matching

```
1: procedure EXPAND(Vertex: v)
2:     for w ∈ N(v) \ C do
3:         if w is M-saturated then
4:             C ← C ∪ {w}
5:             Insert w in Q_s
6:         else if w ∉ W then
7:             W ← W ∪ {w}
8:             Insert w in Q_n
9: procedure TRYTOCONNECT(Vertex: v)
10:    if ∃w ∈ N(v) \ C then
11:        u ← vertex matched with w in M
12:        M ← M \ {uw} ∪ {vw}
13:        C ← C ∪ {v, w}
14:        W ← W \ {v}
15:        Insert v and w in Q_s
16: #Input: G: Graph, M: Maximum matching of G
17: W ← ∅;   Q_s ← ∅;   Q_n ← ∅;
18: r ← any M-saturated vertex
19: C ← {r}
20: Insert r in Q_s
21: while |C| < 2|M| do
22:     while Q_s is not empty do
23:         v ← vertex removed from the front of Q_s
24:         EXPAND(v)
25:     while Q_n is not empty do
26:         v ← vertex removed from the front of Q_n
27:         TRYTOCONNECT(v)
28: return M
```

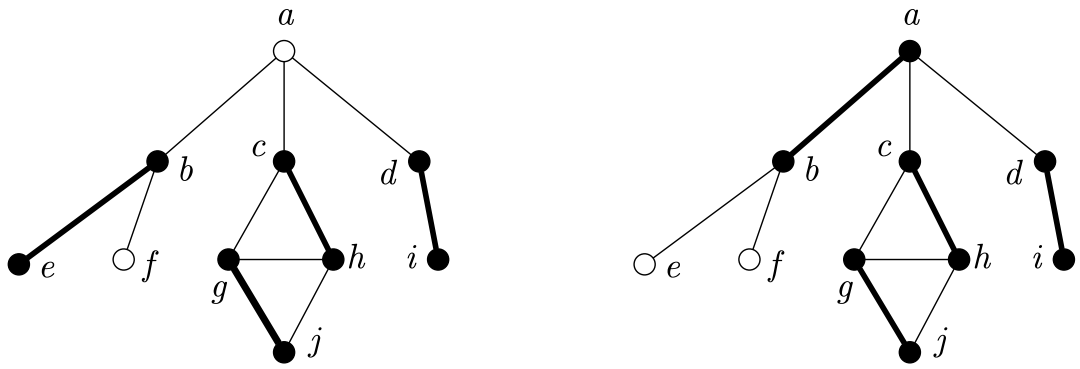


Figure 21 – A maximum matching (a) and a maximum connected matching (b).

Table 7 – Example of variable contents from an execution of the Algorithm 1

Function executed	C	W	Q_s	Q_n	M
Initialization	$\{c\}$	$\{\}$	$\{c\}$	$\{\}$	$\{ch, gj, be, di\}$
EXPAND(c)	$\{c, g, h\}$	$\{a\}$	$\{g,h\}$	$\{a\}$	$\{ch, gj, be, di\}$
EXPAND(g)	$\{c, g, h, j\}$	$\{a\}$	$\{h,j\}$	$\{a\}$	$\{ch, gj, be, di\}$
EXPAND(h)	$\{c, g, h, j\}$	$\{a\}$	$\{j\}$	$\{a\}$	$\{ch, gj, be, di\}$
EXPAND(j)	$\{c, g, h, j\}$	$\{a\}$	$\{\}$	$\{a\}$	$\{ch, gj, be, di\}$
TRYTOCONNECT(a)	$\{c, g, h, j, a, b\}$	$\{\}$	$\{a,b\}$	$\{\}$	$\{ch, gj, ab, di\}$
EXPAND(a)	$\{c, g, h, j, a, b, d\}$	$\{\}$	$\{b,d\}$	$\{\}$	$\{ch, gj, ab, di\}$
EXPAND(b)	$\{c, g, h, j, a, b, d\}$	$\{e\}$	$\{d\}$	$\{e\}$	$\{ch, gj, ab, di\}$
EXPAND(d)	$\{c, g, h, j, a, b, d, i\}$	$\{e\}$	$\{i\}$	$\{e\}$	$\{ch, gj, ab, di\}$
EXPAND(i)	$\{c, g, h, j, a, b, d, i\}$	$\{e\}$	$\{\}$	$\{e\}$	$\{ch, gj, ab, di\}$
TRYTOCONNECT(e)	$\{c, g, h, j, a, b, d, i\}$	$\{e\}$	$\{\}$	$\{\}$	$\{ch, gj, ab, di\}$

Theorem 20. Algorithm 1 runs in $\mathcal{O}(|V| + |E|)$.

Proof. The main loop runs until the connected component has the desired size. At the beginning of each iteration, we know that $G[M]$ is not connected yet because of the loop stopping condition. Since, at the end of the loop, at least one vertex is added to C, the number of iterations is $\mathcal{O}(n)$. The sets C and W act as markers for the queues, so each vertex $v \in V(G)$ can only be processed at most once in the functions EXPAND(v) and TRYTOCONNECT(v). Note that the execution of these functions analyses the neighborhood of v and, therefore, has linear complexity to the degree of v. Hence, the complete algorithm has linear complexity with respect to vertices and edges. □

Table 7 shows an example of the execution of Algorithm 1 with the input matching $M = \{ch, gj, be, di\}$. The vertex c was chosen as the initial. Each line represents the state of the variables after the execution of the function specified in the first column. The lines are in ascending order according to when the functions are executed. The algorithm outputs the maximum connected matching $\{ch, gj, ab, di\}$.

3.2 Weighted Connected Matchings

The problem of deciding whether or not an unweighted graph admits a $\mathscr{P}$-matching of a given size has been investigated for many different properties $\mathscr{P}$. One of the few exceptions of a $\mathscr{P}$-matching problem that is solvable in polynomial time is Connected Matching, as we stated in the previous section. So, it seems reasonable to ask if the problem remains tractable if we generalize it in some form.

Taking into account that weighted variations of matchings are widely studied (see Section 1.2), we decided to consider connected matchings under this approach. Besides, we see that some $\mathscr{P}$-matching concepts were already extended to edge-weighted problems, where, in addition to the matching to have a certain property $\mathscr{P}$, the sum of the weights of the matching edges must be sufficiently large. It was shown that Maximum Weight Induced Matching can be solved in linear time for convex bipartite graphs [115] and in polynomial time for circular-convex and triad-convex bipartite graphs [150]. Fürst et al. showed that Maximum Weight Acyclic Matching is solvable in polynomial time for P_4-free graphs and $2P_3$-free graphs [63].

Sometimes, we approach Weighted Connected Matching separately when negative weights are allowed or not. Note that, unlike some weighted matching problems, such as Maximum Weight Matching, negative weighted edges are relevant and may even be required to be in an optimal solution. As an example, see Figure 22 in which the maximum connected matching in the graph has to contain the negative-weight edge cd.

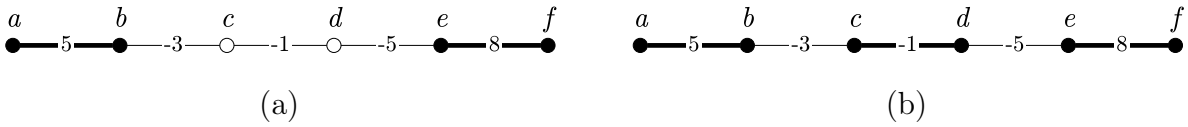


Figure 22 – A maximum matching (a) and a maximum connected matching (b).

Our investigation begins with starlike graphs, a subclass of chordal graphs, where we show that the problem is NP-complete when arbitrary weights are allowed. On the other hand, we present a polynomial-time algorithm for chordal graphs if all weights are non-negative.

Afterwards, we turn our attention to bipartite graphs, first showing that even for non-negative weights, Weighted Connected Matching remains NP-complete, even if the graph is planar or its diameter is bounded. If negative weights are allowed, we prove that the problem remains hard for subcubic planar graphs.

We then prove the existence of polynomial-time algorithms for: (i) graphs of maximum degree two, establishing a complexity dichotomy based on the maximum degree, (ii) graphs of bounded treewidth, and (iii) trees; the latter is an improvement upon the treewidth algorithm, since it runs in linear time. The treewidth algorithm implies fixed-parameter tractability for the treewidth parameterization, which leads us to our final result, where we prove that no polynomial kernel exists when parameterizing by vertex cover unless NP $\subseteq$ coNP/poly.

3.2.1 Chordal graphs

In this section, we focus our study on chordal graphs and one of their subclasses, starlike graphs. Weighted Connected Matching for these classes has different time complexities depending on the admittance of negative weights on the input graph; more

specifically, we show that the problem is NP-complete if negative weights are allowed and is in P otherwise. To reach this result, we prove the NP-completeness for starlike graphs having weights in $\{-1, +1\}$ in Section 3.2.1.1 and a polynomial-time algorithm for chordal graphs having non-negative weights in Section 3.2.1.2.

3.2.1.1 Starlike graphs

Our result is a proof that Weighted Connected Matching is NP-complete on starlike graphs having edge weights in $\{-1, +1\}$. Our reduction is from the NP-complete problem 3SAT [78], with input given by the pair $(X,\mathcal{C})$, where X is the set of variables and $\mathcal{C}$ is the set of clauses; w.l.o.g. we assume that each clause contains exactly three literals. We set the input of Weighted Connected Matching as $k = |X| + |\mathcal{C}|$ and $G_{X,\mathcal{C}}$ built by the following rules:

(I) For each variable $x_i \in X$, add a copy of C_3 whose vertices are labeled x_i, x_i^+ and x_i^-. Set weight -1 to edge $x_i^+x_i^-$ and $+1$ to the other edges.

(II) For each pair of variables x_i,$x_j \in X$, add all possible edges between vertices of $\{x_i^-,x_i^+\}$ and $\{x_j^-,x_j^+\}$ and set its weights to -1.

(III) For each clause $C_i \in \mathcal{C}$, add a copy of K_2 whose edge weight is $+1$ and label its endpoints as c_i^+, and c_i^-. Also, for each literal x_j of C_i, connect both c_i^- and c_i^+ to x_j^- if x is negated, or x_j^+ otherwise; in both cases, the added edges have weight -1.

This graph is indeed starlike, as its clique tree is a star, having as center the maximal clique containing the vertices $\{x_i^+, x_i^- \mid x_i \in X\}$, one leaf clique $\{x_i, x_i^+, x_i^-\}$ for each $x_i \in X$, and one leaf clique for each clause. A connected matching of weight at least k in $G_{X,\mathcal{C}}$ corresponds to an assignment of X such that $x_i \in X$ is set to true if and only if x_i^+ is M-saturated.

Lemma 17. Given a solution for the 3SAT instance $(X,\mathcal{C})$, we can obtain a connected matching M in $G_{X,\mathcal{C}}$ having weight $|X| + |\mathcal{C}|$.

Proof. We show how to obtain the matching M. (i) For each clause $C_i \in \mathcal{C}$, add the edge $c_i^-c_i^+$ to M. Also, (ii) for each variable $x_i \in X$, if x_i is set to true, we saturate the edge $x_i^+x_i$; otherwise, $x_i^-x_i$.

Next, we show that M is connected. Edges from (ii) are connected as each one is incident to a vertex x_i, which is part of the center clique. Each edge from (i), obtained by clause $C_i \in \mathcal{C}$, having x_j as the variable related to the literal that resolves to true in C_j, is connected. This holds because, if x_j is negated, then $c_i^+x_j^- \in E(G_{X,\mathcal{C}})$ and x_j^- is saturated. Otherwise, $c_i^+x_j^+ \in E(G_{X,\mathcal{C}})$ and x_j^+ is saturated. □

Lemma 18. Given an input $(X,\mathcal{C})$ for 3SAT and a connected matching M in $G_{X,\mathcal{C}}$ having weight $|X| + |\mathcal{C}|$, we can obtain an assignment R of X that solves 3SAT.

Proof. Denote by W_{-1} and W_1 the edge sets from $G_{X,\mathcal{C}}$ whose weights are, respectively, -1 and 1. First, we show that a matching having weight $|X| + |\mathcal{C}|$ contains exactly $|X| + |\mathcal{C}|$ edges from W_1 and no edges from W_{-1}.

Note that there can be at most $|X| + |\mathcal{C}|$ edges from W_1. This holds because, for each variable $x_i \in X$, a matching contains at most one edge of $\{x_i^+x_i, x_i^-x_i\}$, since both

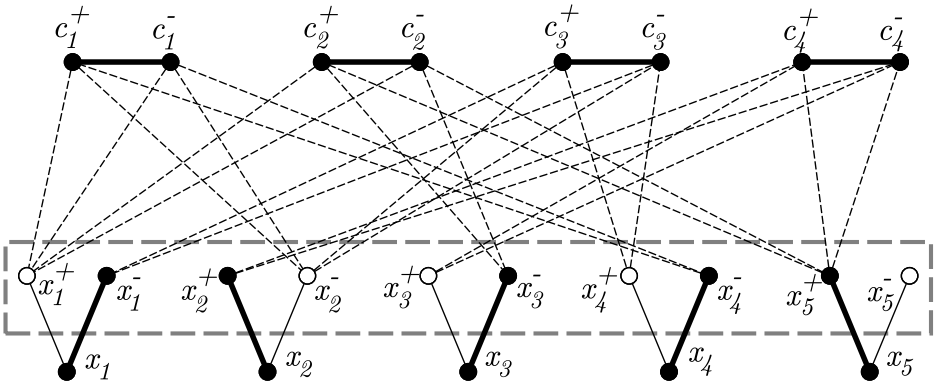


Figure 23 – Example of a starlike graph built by the reduction and a connected matching. Dashed edges have weight -1 and solid edges have weight 1. Vertices in the dashed rectangle induce a clique in which the omitted edges have weight -1.

have an endpoint in vertex x_i. Also, for each clause $C_i \in \mathcal{C}$, the edge $c_i^- c_i^+$ can also be contained in the matching.

Since all edges contained in W_{-1} have negative weights, if there is a matching with $|X| + |\mathcal{C}|$ vertices from W_1 and no vertices from W_{-1}, then it is maximum. Since M is a matching whose weight is $|X|+|\mathcal{C}|$, then $|M \cap W_1| = |X|+|\mathcal{C}|$ and $|M \cap W_{-1}| = 0$. Also, as M is connected, then, for each edge $c_i^+ c_i^- \in M$, there is a saturated vertex adjacent to c_i^+, either x_j^+ or x_j^-, for $x_j \in C_i$ and $C_i \in \mathcal{C}$. Those vertices are exactly the ones representing a literal in clause C_i. So, to obtain R, for each variable $x_i \in X$, we set x_i to true if and only if x_i^+ is saturated. □

Theorem 21. Weighted Connected Matching is NP-complete even for starlike graphs whose edge weights are in $\{-1, +1\}$.

Proof. Note that the problem is in NP. According to the correspondence between Weighted Connected Matching and 3SAT solutions described in Lemmas 17 and 18, the 3SAT problem, which is NP-complete, can be reduced to Weighted Connected Matching using a starlike graph whose edge weights are either -1 or $+1$. Therefore, Weighted Connected Matching is NP-complete even for starlike graphs whose weights are in $\{-1, +1\}$. □

As an example, let an input of 3SAT be $(x_1 \vee \overline{x_2} \vee \overline{x_4}) \wedge (x_1 \vee \overline{x_3} \vee x_5) \wedge (\overline{x_1} \vee \overline{x_2} \vee x_4) \wedge (x_2 \vee x_3 \vee x_5)$. The related graph is illustrated in Figure 23, as well as a connected matching having weight $|X| + |\mathcal{C}| = 9$, corresponding to the assignment (F,T,F,F,T) of the variables $(x_1, x_2, x_3, x_4, x_5)$, in this order.

3.2.1.2 Chordal graphs for non-negative weights

Theorem 21 directly implies that Weighted Connected Matching is also NP-complete for chordal graphs. Interestingly, if the weights are restricted to be non-negative, we can solve Weighted Connected Matching in polynomial time in this class. To this end, we present a polynomial-time reduction to the Maximum Weight Perfect Matching problem, which can be solved in polynomial time [45, 50]. In this problem,

we are given a graph G and we want to find a perfect matching M whose sum of the edge weights is maximum.

For the reduction, we consider $G = (V, E)$ as the input graph to Maximum Weight Connected Matching. We build the input graph $G_p = (V_p, E_p)$ to Maximum Weight Perfect Matching as follows:

(I) Set $V_p = V$. If $|V|$ is odd, add a vertex h to V_p.

(II) Set $E_p = E$. Now, for each pair of non articulation vertices $v_1, v_2 \in V \cup \{h\}$, if $v_1v_2 \notin E$, add the 0-weight edge v_1v_2 to E_p.

In the next lemma, we show that there is always a maximum weight connected matching that saturates all articulations of a graph.

Lemma 19. Let G be a connected graph having no negative weight edge. There is a maximum weight connected matching M that saturates all articulations.

Proof. Let M be a maximum weight connected matching such that an articulation v is not saturated. We show that we can saturate v only by adding edges to M, in a way that M is still connected. Let $\mathcal{C} = \{C_1, \ldots, C_{|\mathcal{C}|}\}$ be the connected components of $G - v$. Note that $|\mathcal{C}| \geq 2$ and the edges of M are contained in exactly one component $C_i \in \mathcal{C}$, because, otherwise, $G[M]$ would not be connected. Let $U = ((\bigcup_{u \in V(M)} N(u)) \setminus V(M)) \setminus \{v\}$. Note that $U \subseteq V(C_i)$. If $U \neq \emptyset$ and there is a path $P = (p_1, \ldots, p_q)$ between $p_1 \in U$ and $p_q = v$ in $G - V(M)$, saturate p_ip_{i+1} for every possible i even. If q is odd, then v is saturated. If P does not exist, or q is even or $U = \emptyset$, saturate v with any vertex of $C_j \neq C_i$ of $\mathcal{C}$. □

Without loss of generality, there is also a maximum weight matching in a chordal graph that saturates all articulations. Therefore, if M_p is a maximum weight perfect matching in G_p, we can obtain in linear time a maximum weight connected matching which saturates all articulations in G by the union of $M^* = M_p \cap E$ with a maximal set S of 0-weight edges having endpoints in vertices not saturated by M^*. This results in a connected matching, as otherwise we would find a minimal separator of G that has at least two non-saturated vertices; since G is chordal, these vertices are adjacent, which contradicts the maximality of S.

For this purpose, we prove in Lemma 20 that G_p has a perfect matching. Then, Lemma 21 shows that we can find a maximum weight connected matching in G given a maximum weight perfect matching in G_p. Theorem 22 finishes the proof, proving that Weighted Connected Matching can be solved in polynomial time for chordal graphs.

Lemma 20. The graph G_p has a perfect matching.

Proof. Let T be a block-cutpoint tree of G rooted in a vertex denoted by r. Also, denote A by the set of articulations of G. Next, we iteratively build a perfect matching M in G_p. First, we saturate all vertices of A. Starting from r, we traverse T^r in preorder and, when visiting a non saturated vertex $v \in A$, we saturate v by the following way. If v has a non leaf child B_i, we saturate v with any child of B_i, that is contained in A. Otherwise, let B_i be one child of v. Then, we saturate v with any vertex u contained in the block of G represented by B_i.

As an example, in Figure 24, consider the graph and its block-cutpoint tree rooted in B_4. The matched edges by this procedure that are incident to articulations, in order, can be c_3c_2, c_1v, c_5c_7 and c_4c_6.

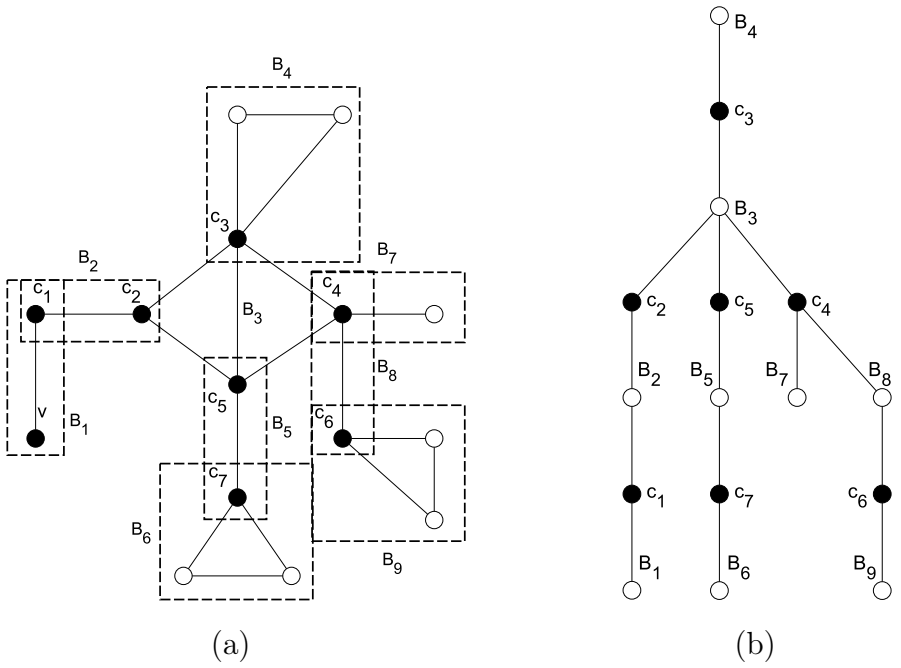


Figure 24 – A chordal graph in (a) and its block-cutpoint tree in (b).

Note that, when visiting a non saturated vertex $v \in A$, none of its descendant vertices are saturated. Moreover, v is not a leaf, because, as it is an articulation, it is contained in at least two blocks of G. At least one of these blocks is a child of v in T^r. Thus, there will always be at least one vertex to saturate with v.

Finally, we saturate all the remaining vertices, of $V_p \setminus A$, which induce a clique. After that, we saturated every vertex of V_p. Then, G_p has a perfect matching. □

Lemma 21. Given a maximum weight perfect matching M_p in G_p, we can obtain, in linear time, a maximum weight connected matching M of G.

Proof. We build a maximum weight connected matching M in G. First, (i) we set $M = M_p \cap E(G)$. At this point, there may be vertices in V not M-saturated and connected by 0-weight edges. Finally, (ii) we saturate a maximal set of those vertices. Note that $w(M) = w(M_p)$ and this can be done in linear time.

Now, we prove that M is connected because, in each separator S of G, there is at least one saturated vertex. Suppose this is not true. Note that $|S| \geq 2$, since all articulations of G are M_p-saturated by edges that are also contained in M. Then, let v_1,$v_2 \in S$ be two vertices not saturated. Since S is a clique, these two vertices had to be saturated in (ii). Then, for each separator S of G, there is at least one saturated vertex, which makes M connected.

Now, we show that M is maximum. Suppose there is a connected matching M' with greater weight than M. Moreover, based on Lemma 19, we assume that M' saturates all articulations. So, we build a perfect matching M'_p from M'. First, set $M'_p = M'$. Since all articulations of G are saturated by M', then, all the remaining vertices induce a clique. Finally, we saturate all those vertices by any disjoint set of edges. Observe that $w(M'_p) \geq w(M') > w(M_p) = w(M)$, which is a contradiction because M_p is a maximum weight perfect matching in G_p. Therefore the matching M is a maximum weight connected matching in G, and it can also be obtained in linear time. □

Theorem 22. Maximum Weight Connected Matching for chordal graphs whose edge weights are all non-negative can be solved in polynomial time.

Proof. To solve Maximum Weight Connected Matching for an input graph G, we can build, in linear time, the graph G_p and use it as an input of a Maximum Weight Perfect Matching algorithm. Given this answer, we obtain a maximum weight connected matching of G in linear time, as stated in Lemma 21. Therefore, Maximum Weight Connected Matching can be solved in polynomial time, with the same time complexity as Maximum Weight Perfect Matching. □

3.2.1.3 Example of a chordal graph reduction

Let the input graph G of Maximum Weight Connected Matching be the one illustrated in Figure 25a, where we show a maximum weight matching, whose weight is 44. Observe that this matching is not connected as f is not saturated. Conversely, in Figure 25b, we show a maximum weight perfect matching M_p of G_p, whose weight is 42. To simplify the figure, some 0-weight edges are omitted. This matching is used to obtain the maximum weight connected matching in G of Figure 25c by removing the edge lm.

3.2.2 Bipartite graphs

The hardness result stated in Theorem 21 is highly dependent on the existence of several non-trivial cliques, which are forbidden in some graph classes, including bipartite graphs. This raises the question of whether or not the absence of these structures makes the problem easier. In this section, we answer this in the negative by showing that Weighted Connected Matching remains hard on bipartite graphs having only binary weights.

Our proof is also based on a reduction from 3SAT, whose input is $(X,\mathcal{C})$. Let the input of Weighted Connected Matching be $k = |X| + |\mathcal{C}| + 1$ and the graph $G_{X,\mathcal{C}}$ obtained by the following rules.

(I) Add two vertices, h^+ and h^-, connected by a 1-weight edge.

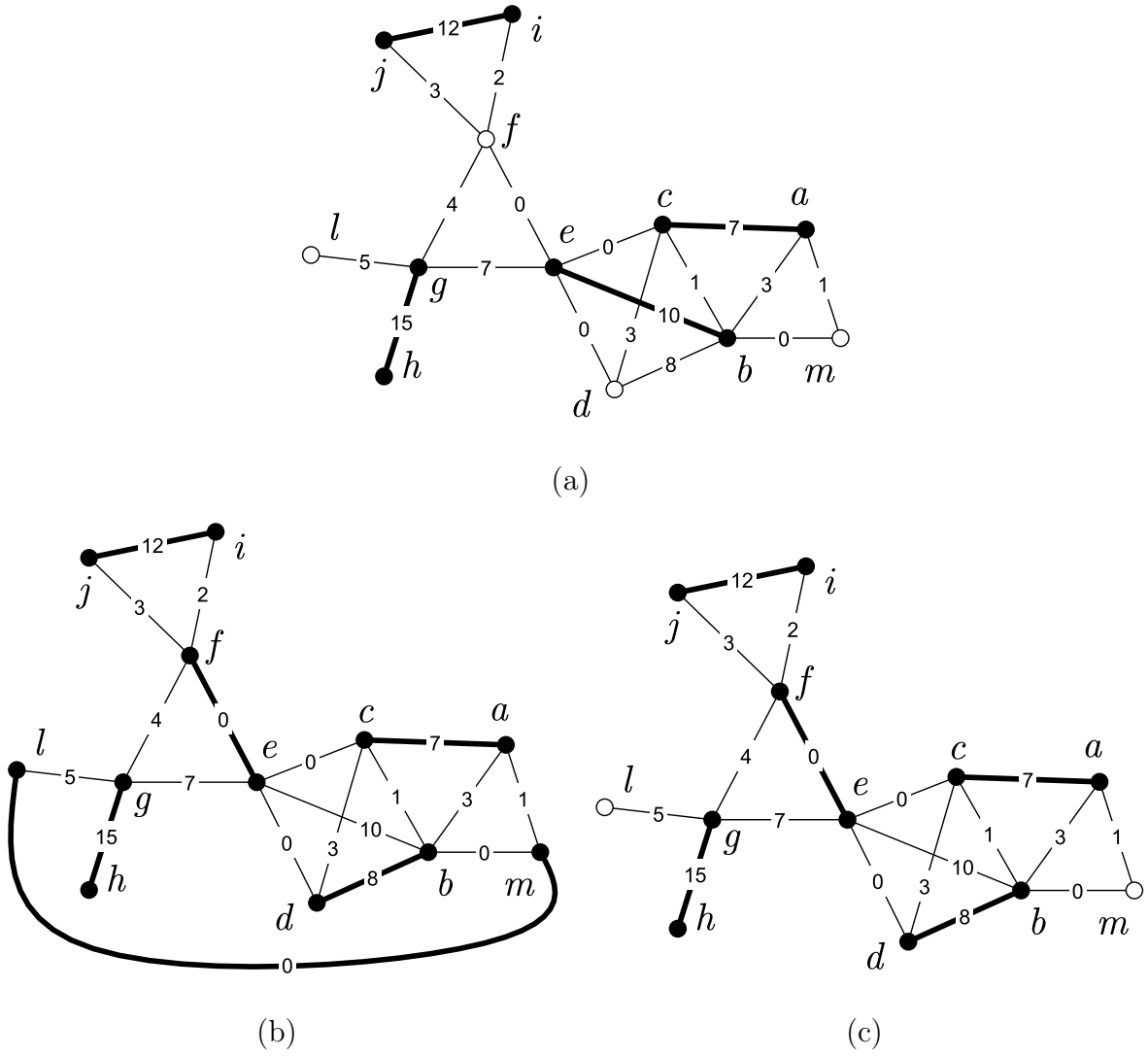
(II) For each variable $x_i \in X$, add a copy of P_3 whose edge weights are 1, and label its endpoints as x_i^+ and x_i^-. Moreover, connect the other vertex, labeled x_i, to h^+ and set this edge weight to 0.

(III) For each clause $C_i \in \mathcal{C}$, add a copy of K_2 whose edge weight is 1 and label its vertices as c_i^+ and c_i^-. Also, for each literal x_j of C_i, add a 0-weight edge $c_i^+x_j^-$ if x_j is negated, or $c_i^+x_j^+$ otherwise.

A connected matching M of weight at least k of $G_{X,\mathcal{C}}$ corresponds to an assignment of X such that variable x_i is set to true if and only if x_i^+ is M-saturated.

Figure 26 presents an example where the input formula of 3SAT is $(x_1 \vee \overline{x_2} \vee \overline{x_4}) \wedge (x_1 \vee \overline{x_3} \vee x_5) \wedge (\overline{x_1} \vee \overline{x_2} \vee x_4) \wedge (x_2 \vee x_3 \vee x_5)$. The illustrated connected matching corresponds to the assignment (F,T,F,F,T), in this order, of the variables $(x_1, x_2, x_3, x_4, x_5)$.

It is also possible to strengthen our NP-completeness proof in terms of the graph diameter, adding a vertex u and the 0-weight edges defined by $\{c_i^+u \mid C_i \in \mathcal{C}\} \cup \{x_i^+h^-, x_i^-h^- \mid x_i \in X\} \cup \{x_ix_j^+, x_ix_j^-, x_jx_i^+, x_jx_i^- \mid x_i,x_j \in X\}$. Connected matchings of weight k have the same properties and patterns as previously stated. In Figure 27, we



(a) (b) (c)

Figure 25 – A chordal graph G and its maximum weight matching in (a), a maximum weight perfect matching of G_p in (b) and its corresponding maximum weight connected matching of G in (c).

illustrate an example with a full diameter four bipartite graph, generated in our reduction from the 3SAT input $B = (x_1 \vee \overline{x_2} \vee \overline{x_4}) \wedge (x_1 \vee \overline{x_3} \vee \overline{x_4}) \wedge (\overline{x_1} \vee \overline{x_2} \vee x_4) \wedge (x_2 \vee \overline{x_3} \vee \overline{x_4})$. Dashed and solid edges represent, respectively, edges having weights 0 and 1.

In next lemmas, we show that, given an input $(X,\mathcal{C})$ for 3SAT, it is possible to obtain in linear time a connected matching M in $G_{X,\mathcal{C}}$, $w(M) = k$, if we have a solution for $(X,\mathcal{C})$, and vice versa. Denote W_0 and W_1 by the edge sets from $G_{X,\mathcal{C}}$ whose weights are, respectively, 0 and 1.

Lemma 22. Given a solution for the 3SAT instance $(X,\mathcal{C})$, we can obtain a connected matching M having weight $|X| + |\mathcal{C}| + 1$ in $G_{X,\mathcal{C}}$.

Proof. We show how to obtain the matching M. (i) For each clause $C_i \in \mathcal{C}$, add the edge $c_i^- c_i^+$ to M. Also, (ii) for each variable $x_i \in X$, if x_i is set to true, we saturate the edge $x_i^+ x_i$; otherwise, $x_i^- x_i$. Moreover, (iii) saturate the edge $h^+ h^-$.

Now, we prove that this matching is connected. Edges from (ii) are connected as they are connected to the edge $h^+ h^-$ of (iii). Each edge from (i), obtained by clause $C_i \in \mathcal{C}$,

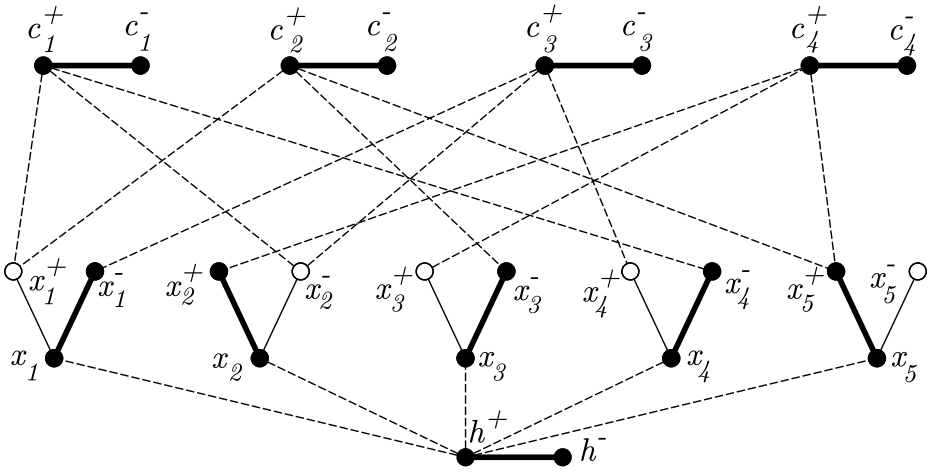


Figure 26 – Example of a bipartite graph built by the reduction and a connected matching of size $|X| + |\mathcal{C}| + 1 = 10$. Dashed and solid edges represent weights 0 and 1, respectively.

having x_j as the variable related to a literal that resolves to true in C_i, is connected. This holds because, if x_j is negated, then $c_i^+ x_j^- \in E(G_{X,\mathcal{C}})$ and x_j^- is saturated. Otherwise, $c_i^+ x_j^+ \in E(G_{X,\mathcal{C}})$ and x_j^+ is saturated. □

Lemma 23. Given an input $(X,\mathcal{C})$ for 3SAT and a connected matching M in $G_{X,\mathcal{C}}$ having weight $|X| + |\mathcal{C}| + 1$, we can obtain an assignment of X that solves 3SAT in polynomial time.

Proof. First, we show that a matching M having weight $|X| + |\mathcal{C}| + 1$ contains exactly $|X| + |\mathcal{C}| + 1$ edges from W_1 and no edges from W_0. Note that $|M \cap W_1| \leq |X| + |\mathcal{C}| + 1$ because, for each variable $x_i \in X$, there is at most one matched edge of $\{x_i^+ x_i, x_i^- x_i\}$, since both have an endpoint in vertex x_i. Also, M can contain the edges $\{h^+h^-\} \cup \{c_i^- c_i^+ \mid C_i \in \mathcal{C}\}$.

Observe that each edge from W_0 has an endpoint in either $\{h^+\}$ or $S_1 = \{c_i^+ \mid C_i \in \mathcal{C}\}$. Saturating any of these vertices by a W_0 edge will decrease the number of possibly matched edges of W_1 and the weight of M, resulting in $w(M) < k$. Namely, if we saturate h^+ or $c_i^+ \in S_1$, we are not able to saturate, respectively, h^+h^- or $c_i^+ c_i^-$. Thus, $|M \cap W_1| = |X| + |\mathcal{C}| + 1$ and $|M \cap W_0| = 0$.

If the matching M is connected, then, for each edge $c_i^+ c_i^- \in M$, there is a M-saturated vertex adjacent to c_i^+, either x_j^+ or x_j^-, for a variable x_j contained in clause C_i. Also, for each variable $x_i \in X$, the vertex x_i is saturated, which is connected to the edge h^+h^-. Hence, to obtain an assignment to X, for each variable $x_i \in X$, we set x_i to true if and only if x_i^+ is saturated. □

Theorem 23. Weighted Connected Matching is NP-complete on bipartite graphs of diameter 4 even if all edge weights are in {0,1}.

Proof. Note that the problem is in NP. According to the correspondence between Weighted Connected Matching and 3SAT solutions described in Lemmas 22 and 23, 3SAT, which is NP-complete, can be reduced to Weighted Connected Matching using a bipartite graph whose diameter is 4 and the edge weights are either 0 or 1. Hence,

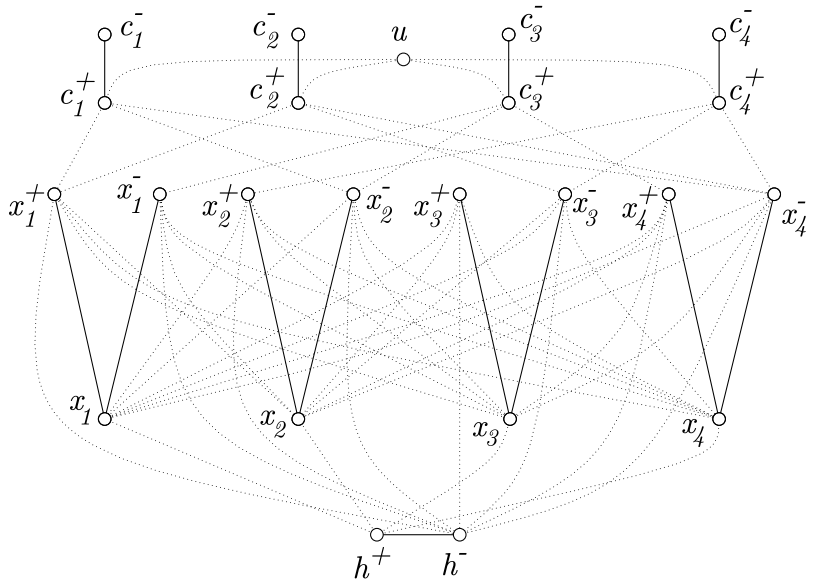


Figure 27 – Example of a diameter 4 bipartite graph built by the reduction.

Weighted Connected Matching is NP-complete even for bipartite graphs whose weights are in $\{0, 1\}$ and diameter is 4. □

A follow-up question is if there are subclasses of bipartite graphs that admit polynomial-time algorithms. As we show in Section 3.2.4, there exist such algorithms for trees and graphs of maximum degree two. Nevertheless, we would like to study non-trivial classes, such as chordal bipartite graphs and planar bipartite graphs. We leave the former as an open problem, but proceed to study the latter, and other subclasses of planar graphs, in our next section.

3.2.3 Planar graphs

Aside from planar bipartite graphs, which is shown to be NP-complete in Section 3.2.3.1, we investigate the complexity of Weighted Connected Matching in planar graphs under degree constraints in Section 3.2.3.3, proving that the problem remains hard in the subcubic case.

3.2.3.1 Planar bipartite graphs

In this section, we prove the NP-completeness of Weighted Connected Matching on planar bipartite graphs having weights of either 0 or 1. We use a polynomial-time reduction from an SAT variant that we explain in the following.

Let B be a conjunctive formula where $\mathcal{C} = \{C_1, \ldots, C_q\}$ and $X = \{x_1, \ldots, x_m\}$ are the sets of clauses and variables of B, respectively. Let $X_c = (x_1, \ldots, x_m)$ be an ordering of X. Let $G(B) = (V,E)$ be the graph in which there is a vertex for each clause and each variable of B, namely $V = \{x_i \mid x_i \in X\} \cup \{c_j \mid C_j \in \mathcal{C}\}$. The edge set E is partitioned in A_1, A_2 where $A_1 = \{x_i c_j \mid \{x_i, \overline{x_i}\} \cap C_j \neq \emptyset\}$ and $A_2 = \{x_i x_{i+1} \mid 1 \leq i < m\} \cup \{x_m x_1\}$. Note that A_2 induces a cycle containing all variable vertices and follows the ordering X_c.

In [125], Lichtenstein defined a conjunctive formula B as planar if $G(B)$ is planar; he showed that, for every instance of SAT, we can build in polynomial time an equivalent

planar formula as well as its planar embedding. The problem of finding a true assignment to a planar formula was named as PLANAR SAT, and was proven to be NP-complete. Later in the paper, the author also proved the NP-completeness of a PLANAR SAT variant of our interest. In this problem, the input planar boolean formula B is monotone, that is, each of its clauses consists only of positive literals or only of negative literals. Also, there is a planar embedding of $G(B)$ such that each edge referencing a positive(negative) literal is connected to the top(bottom) of the variable vertices. We refer to it as PLANAR MONOTONE SAT. We assume that X_c is also part of its input, since it was shown in [125] how to obtain such embedding of $G(B)$ in polynomial time.

We use a reduction from PLANAR MONOTONE SAT to WEIGHTED CONNECTED MATCHING, for which the input is defined as $k = 2|X| + |\mathcal{C}|$ and the planar bipartite graph, which we call $H(B)$, obtained from $G(B)$, with the addition of the following rules.

(I) For each variable $x_i \in X$, generate four vertices, x_i^+, x_i^-, v_i, u_i. Also, add the 1-weight edges $\{v_iu_i, x_ix_i^+, x_ix_i^-\}$.

(II) For each edge having an endpoint in x_i, $1 \leq i \leq |X|$, representing a positive(negative) literal, we connect this edge to $x_i^+(x_i^-)$ instead of x_i and set its weight to 1.

(III) Remove all the edges from the variable vertices cycle, that is, from A_2, and add the 0-weight edges $\{v_ix_i, v_ix_{i+1} \mid 1 \leq i < |X|\} \cup \{v_mx_m, v_mx_1\}$.

(IV) For each clause $C_i \in \mathcal{C}$, we label the corresponding vertex as c_i^+ and connect it to a new vertex, c_i^-, by a 1-weight edge.

Similarly to the reduction described in Section 3.2.2, a connected matching M having weight at least k in $H(B)$ corresponds to an assignment of B in such a way that variable $x_i \in X$ is set to true if and only if x_i^+ is M-saturated. This is proved in Lemmas 25 and 26, preceded by Lemma 24, which demonstrates properties of $H(B)$. Then, Theorem 24 concludes the NP-completeness proof.

Lemma 24. The graph $H(B)$ is planar and bipartite.

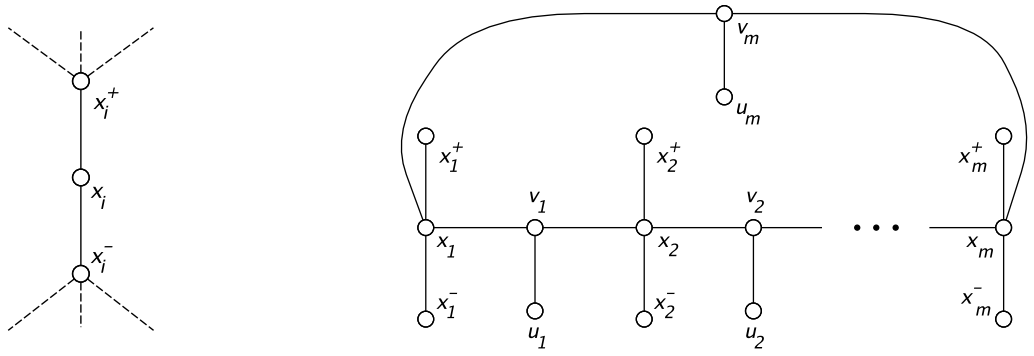
Proof. Observe that $H(B)$ is bipartite since one of its bipartitions can be defined as $\{x_i,u_i \mid 1 \leq i \leq |X|\} \cup \{c_i^+ \mid C_i \in \mathcal{C}\}$. Next, we show that $H(B)$ is planar. Note that the former graph $G(B)$ has all edges connecting positive(negative) literals are connected to the top(bottom) of the variable vertices.

For each variable $x_i \in X$ we created vertices x_i^+ and x_i^- representing the literals x_i and $\overline{x_i}$. Then, in the graph embedding, we can position x_i^+ and x_i^-, respectively, on the top and on the bottom of x_i. Those literal variables do not cross the former variable cycle embedding, as we can see in Figure 28a.

Also, the vertices v_i and u_i can be positioned along the edges of the variable vertices cycle, as in Figure 28b. Therefore, this graph is bipartite and planar. □

Lemma 25. Given a planar formula B and an assignment R that resolves B to true, we can obtain in linear time a connected matching M in $H(B)$ such that $w(M) = k = 2|X| + |\mathcal{C}|$.

Proof. Let's build a connected matching M having weight k. First, for each variable $x_i \in X$, we add to M the edge $x_ix_i^+$ if x_i is true in R and, otherwise, $x_ix_i^-$. At this point, we have $|X|$ saturated 1-weight edges. We also saturate the 1-weight edges of



(a) (b)

Figure 28 – Planar representations of some subgraphs of $H(B)$.

$\{v_iu_i \mid x_i \in X\} \cup \{c_i^+c_i^- \mid C_i \in \mathcal{C}\}$. At this point, we have $2|X| + |\mathcal{C}|$ saturated 1-weight edges in M and, thus, $w(M) = 2|X| + |\mathcal{C}|$.

Note that the matching is connected since, the vertices of $\{x_i, v_i \mid 1 \leq i \leq |X|\}$ induce a cycle in $H(B)$ and are all saturated. The remaining edges that are not connected to those vertices are exactly the ones in $\{c_i^+c_i^- \mid 1 \leq i \leq |\mathcal{C}|\}$. For $1 \leq i \leq |\mathcal{C}|$, the vertex c_i^+ is connected to at least one saturated vertex x_i^+ or x_i^-, corresponding to literals, respectively x_i or $\overline{x_i}$, which resolves to true. This is due to the fact that we saturated the literal vertices according to the assignment R, that resolves B to true. Thus, M is connected. □

Lemma 26. Given the graph $H(B)$ obtained by the formula B, and a connected matching M, $w(M) = 2|X| + |\mathcal{C}|$ in $H(B)$, we can obtain in linear time a variable assignment of X that resolves B to true.

Proof. Denote by W_0 and W_1 the edge sets of $H(B)$ whose weights are, respectively, 0 and 1. First, we show that a matching having weight $2|X|+|\mathcal{C}|$ contains exactly $2|X|+|\mathcal{C}|$ edges from W_1 and no edges from W_0.

First, we show that $M \cap W_1 \leq 2|X| + |\mathcal{C}|$ because, for each variable $x_i \in X$, there is at most one matched edge of $\{x_i^+x_i, x_i^-x_i\}$ since both have an endpoint in vertex x_i. The other edges from W_1, $\{c_i^+c_i^- \mid 1 \leq i \leq |\mathcal{C}|\} \cup \{v_iu_i \mid 1 \leq i \leq |X|\}$ can all be saturated. Conversely, each edge from W_0 has an endpoint in $S_1 = \{c_i^+ \mid 1 \leq i \leq |\mathcal{C}|\}$ or in $S_2 = \{v_i \mid 1 \leq i \leq |X|\}$. Saturating any of these vertices by a W_0 edge will decrease the maximum number of matched edges of W_1 and, thus, decrease the weight of M such that $w(M) < k$. Namely, if we saturate $c_i^+ \in S_1$ or $v_i \in S_2$ by 0-weight edges, M can not contain, respectively, $c_i^+c_i^-$ or v_iu_i. Therefore, $|M \cap W_1| = 2|X| + |\mathcal{C}|$ and $|M \cap W_0| = 0$.

Moreover, M is connected, then, for each matched edge $c_i^+c_i^-$, there is a saturated adjacent vertex, either x_j^+ or x_j^-, $x_j \in X$, for a variable x_j contained in clause C_i. Those vertices are exactly the ones representing literals in the clause c_i that resolves c_i to true. Also, the vertices $\{x_iv_i \in 1 \leq i \leq |X|\}$ are all saturated and connected.

Hence, to obtain an assignment of X, for each variable $x_i \in X$, we set x_i to true if and only if x_i^+ is saturated, which can be done in linear time. □

Theorem 24. Weighted Connected Matching is NP-complete on planar bipartite graphs whose edge weights are in $\{0, 1\}$.

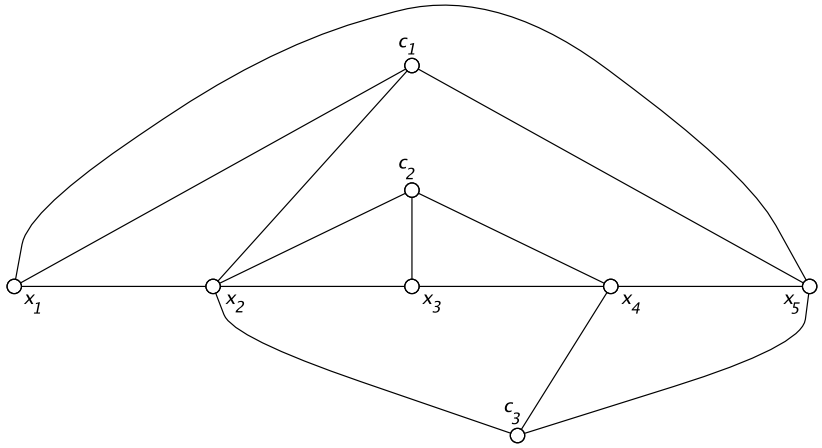


Figure 29 – Example of a graph $G(B)$ generated from formula $B = (x_1 \vee x_2 \vee x_5) \wedge (x_2 \vee x_3 \vee x_4) \wedge (\overline{x_2} \vee \overline{x_4} \vee \overline{x_5})$ and $X_c = (x_1, \ldots, x_5)$.

Proof. Note that the problem is in NP. In Lemmas 25 and 26, we show the correspondence between solutions of Weighted Connected Matching and Planar Monotone SAT, which is NP-complete. Thus, there is a polynomial-time reduction from Planar Monotone SAT to Weighted Connected Matching using a planar and bipartite input graph whose edge weights are either 0 or 1. Therefore, Weighted Connected Matching is NP-complete even for planar bipartite graphs with binary weights. □

3.2.3.2 Example

Let B be an input for Planar Monotone SAT defined as $(x_1 \vee x_2 \vee x_5) \wedge (x_2 \vee x_3 \vee x_4) \wedge (\overline{x_2} \vee \overline{x_4} \vee \overline{x_5})$ and $X_c = (x_1, \ldots, x_5)$. Observe an example of a graph $G(B)$ in Figure 29. This formula is monotone as, in each clause, literals are all either positive or negative. Moreover, $G(B)$ is planar, and all edges representing positive(negative) literals are connected at the top(bottom) of variable vertices.

Thus, B and X_c is a legit input example of Planar Monotone SAT, which corresponds to the input $k = 2|X| + |\mathcal{C}| = 13$ and $H(B)$ as in Figure 30 for Weighted Connected Matching. The connected matching of weight $k = 13$ corresponds to the assignment (T,F,T,T,T) of the variables $(x_1, x_2, x_3, x_4, x_5)$, in this order.

3.2.3.3 Subcubic planar graphs

Now, we approach subcubic planar graphs, showing the NP-completeness of Weighted Connected Matching for this class having edge weights in $\{-1, +1\}$. Our proof is very similar to another reduction, made by Marzio De Biasi [15].

Let us define a polynomial-time reduction from Steiner Tree, one of Karp's original 21 NP-complete problems [111]. In this problem, we are given a graph $H = (V_H, E_H)$, a subset $R \subseteq V_H$ and an integer $k' > 0$. We want to know if there is a subgraph $T = (V_T, E_T)$ of H such that T is a tree, $R \subseteq V_T$ and $|E_T| \leq k'$. Our reduction is from Steiner Tree in which the input graph is planar. Garey and Johnson showed in [77] that this problem is NP-complete.

Let $(H = (V_H, E_H), R, k')$ be the input of Steiner Tree such that H is planar. Also, let $q = \Delta(H)$, $p = q(|V_H| - |R|) + 1$ and $r = p|E_H| + 1$. For Weighted Connected

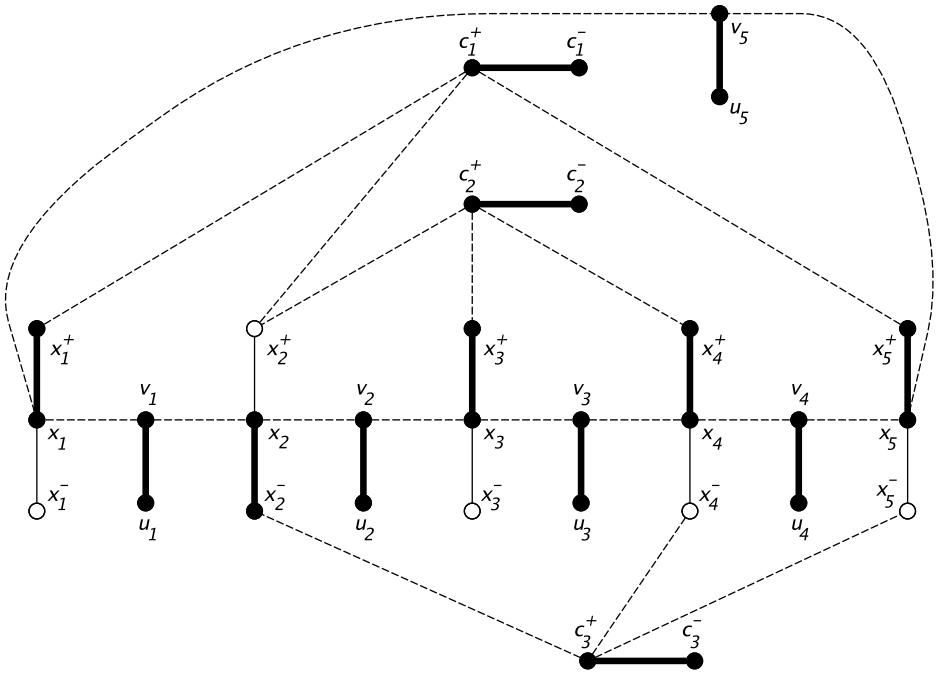


Figure 30 – Example of a graph $H(B)$ and a connected matching of size k generated from the formula $B = (x_1 \vee x_2 \vee x_5) \wedge (x_2 \vee x_3 \vee x_4) \wedge (\overline{x_2} \vee \overline{x_4} \vee \overline{x_5})$ and $X_c = (x_1, \ldots, x_5)$. Dashed and solid edges represent weights 0 and 1, respectively.

MATCHING input (G,k), we define $k = r|R| - pk'$ and the graph G built by the following procedures:

(I) For each vertex $w \in V_H$, add a copy of a cycle with $2r$ vertices if $w \in R$, or $2q$ otherwise. If this number is less than 3, instead, add a copy of a path with the same number of vertices. Set the weights of all these edges to 1. Now, for each $u \in N(w)$, arbitrarily label one of the vertices of this subgraph as v_{wu}. We denote this subgraph by C_w.

(II) For each edge wu in E_H, generate a copy of P_{2p} whose edge weights are -1 and make one of its endpoints vertices adjacent to v_{wu} and the other to v_{uw}. We denote this subgraph by P_{wu} or P_{uw}.

A connected matching of weight at least k in G corresponds to a tree $T = (V_T,E_T)$ where $V_T = \{w \mid w \in V_H, V(C_w) \cap V(M) \neq \emptyset\} \subseteq R$ and $E_T = \{wu \mid wu \in E_H, V(P_{wu}) \subseteq V(M)\}$.

To prove this equivalency, we begin by analyzing the planarity of G, showing that a planar embedding of H can be used to build a planar embedding of G. Note that the cycles in G generated in (I) can be positioned in the same place as vertices in H. Also, all paths in G generated in (II) can be positioned along the edges as in H. Now, we turn our attention to the maximum vertex degree of G. Note that every vertex in the cycles of (I) has degree 2, except the ones connected to one vertex of a path from (II), having degree 3. All vertices from (II) have degree 2. We then conclude that G is both planar and subcubic.

In Lemmas 28 and 27, we show the correspondence between the solutions of STEINER TREE and WEIGHTED CONNECTED MATCHING. Finally, Theorem 25 concludes our

NP-completeness proof.

Lemma 27. Let (H,R,k') be an input of Steiner Tree , whose solution is $T = (V_T,E_T)$, and G be the reduction graph obtained from it. We can obtain, in polynomial time, a connected matching M in G such that $w(M) \geq k = r|R| - pk'$.

Proof. Let us build a connected matching M having a size at least k.

For each vertex $w \in V_T$, we saturate $\frac{|V(C_w)|}{2} = r$ edges from C_w. Note that the sum of the weights of these edges is $r|R|$, since all vertices of R are contained in V_T and, for each cycle C_u, $u \in R$, we can saturate r 1-weight edges.

Moreover, for each edge $uw \in E_T$, we saturate $\frac{|V(P_{uw})|}{2} = p$ edges from P_{uw}. The sum of these edge weights is at most $-pk'$, since, for each P_{uw}, we saturate p edges having weight -1, and $k' \geq |E_T|$.

Next, we show that M is connected. Note that $V(M) = \{V(C_w) \mid w \in V_T\} \cup \{V(P_{uw}) \mid uw \in E_T\}$. So, for each $uw \in E_T$, the vertices $V(C_u) \cup V(C_w) \cup \{v_{uw},v_{wu}\}$ are saturated. Hence, M is connected, $w(M) \geq r|R| - pk'$, and can be obtained in polynomial time. □

Lemma 28. Let $(H = (V_H,E_H),R,k')$ be an input of Steiner Tree, G be the reduction graph obtained from it, and M a connected matching in G such that $w(M) \geq k = r|R| - pk'$. We can obtain in polynomial time a tree $T = (V_T, E_T)$ that is a solution of Steiner Tree instance.

Proof. We show that, in order to $w(M) \leq k$, we have to saturate vertices of every $V(C_w)$, $w \in R$. Suppose that this is not true, so there is a vertex $u \in R$ such that $|V(M) \cap V(C_u)| = 0$. Then, the weight of M is at most the maximum number of 1-weight edges of $C = \{C_w \mid u \neq w \in V_H\}$. For each cycle $C_w \in C$ M contains at most $\frac{|V(C_w)|}{2}$ edges. So, $w(M) \leq \sum_{w \in V_H} \frac{|C_w|}{2}$, and we have the following equation.

$$\sum_{w \in V_H \setminus \{u\}} \frac{|C_w|}{2} = \sum_{w \in R \setminus \{u\}} \frac{|C_w|}{2} + \sum_{w \in V_H \setminus R} \frac{|C_w|}{2} = r(|R| - 1) + q(|V_H| - |R|)$$

If $w(M)$ is at least k, then $r(|R| - 1) + q(|V_H| - |R|) \geq k = r|R| - pk'$.

$$\begin{aligned} r(|R| - 1) + (p - 1) &\geq r|R| \geq r|R| - pk' \\ p - 1 &\geq r \\ p &\geq p|E_H| + 2 \end{aligned}$$

This is a contradiction, which means that, for every $w \in R$, at least one vertex of $V(C_w)$ is saturated. Moreover, M can contain r 1-weight edges from it. As for M to be connected, some $V(P_{uw})$ are all saturated, for $uw \in E_H$. The number of those paths that have all their vertices saturated is at most k', since $w(M) \geq r|R| - pk'$.

Note that saturating vertices of $V(C_w)$, $w \in V_H \setminus R$, is irrelevant because $p = q(|V| - |R|) + 1$, and even if we saturate all those cycles, the maximum weight obtained is $q(|V| - |R|)$.

Thereby, T can be defined as $V_T = \{w \mid w \in V_H, V(C_w) \cap V(M) \neq \emptyset\} \subseteq R$ and $E_T = \{wu \mid wu \in E_H, V(P_{wu}) \subseteq V(M)\}$. Note that $|E_T| \leq k'$. □

Theorem 25. Weighted Connected Matching is NP-complete even for subcubic planar graphs having edge weights in $\{-1, +1\}$.

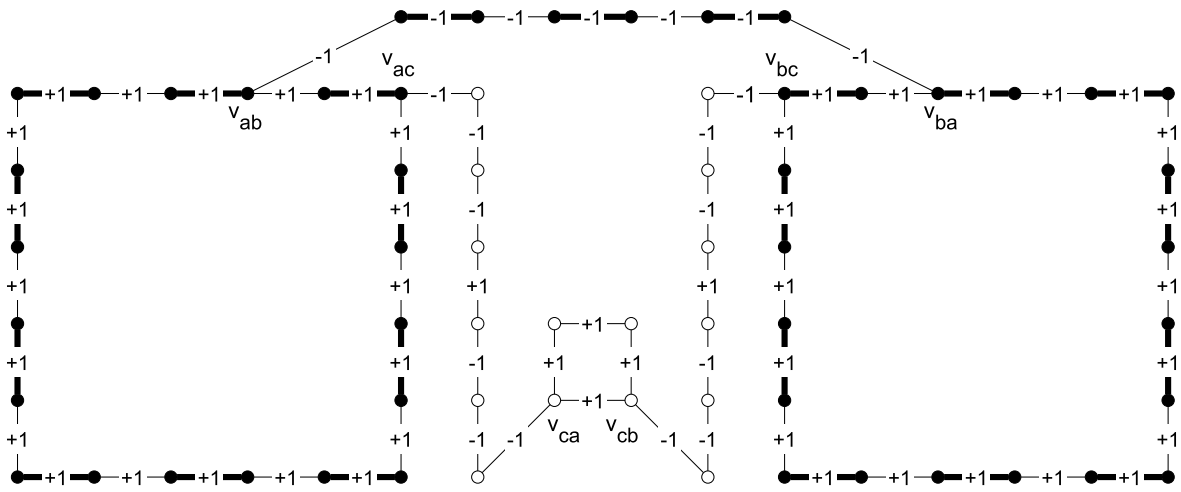


Figure 31 – Example of a subcubic planar graph G generated from a cycle of length three. The connected matching represents the tree subgraph $G[\{a,b\}]$ of G.

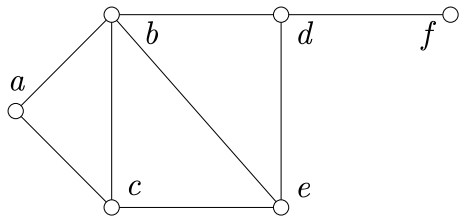


Figure 32 – Example of an input graph for Steiner Tree.

Proof. Note that the problem is in NP. According to the correspondence between Weighted Connected Matching and Steiner Tree solutions that are described in Lemmas 28 and 27, Steiner Tree problem restricted to planar graphs, which is NP-complete, can be reduced to Weighted Connected Matching using a planar graph whose edge weights are either -1 or $+1$ and vertex degree is at most 3. Therefore, Weighted Connected Matching is NP-complete even for subcubic planar graphs whose weights are in $\{-1,+1\}$. □

3.2.3.4 Example 1

As an example, consider the input for Steiner Tree as $k' = 1$, $R = \{a,b\}$ and $H = (V_H, E_H)$, isomorphic to C_3, with $V_H = \{a,b,c\}$. So, $q = \Delta(H) = 2$, $p = q(|V_H| - |R|) + 1 = 3$, $r = p|E_H| + 1 = 10$. For the Weighted Connected Matching input, we set $k = 17$, and the graph as illustrated in Figure 31.

3.2.3.5 Example 2

Let the input for Steiner Tree be $k' = 3$, $R = \{a,c,d\}$ and the graph $H = (V_H,E_H)$ as in Figure 32. In this case, we have that $q = \Delta(H) = 4$, $p = q(|V_H| - |R|) + 1 = 13$, and $r = p|E_H| + 1 = 105$.

A representation of the reduction graph G is shown in Figure 33. For a better visualization, we replace path subgraphs P_{2p} with a square. Each of its edges is incident to terminal vertices of P_{2p}. Also, a triangle represents a cycle subgraph, in which $\triangledown$ is a

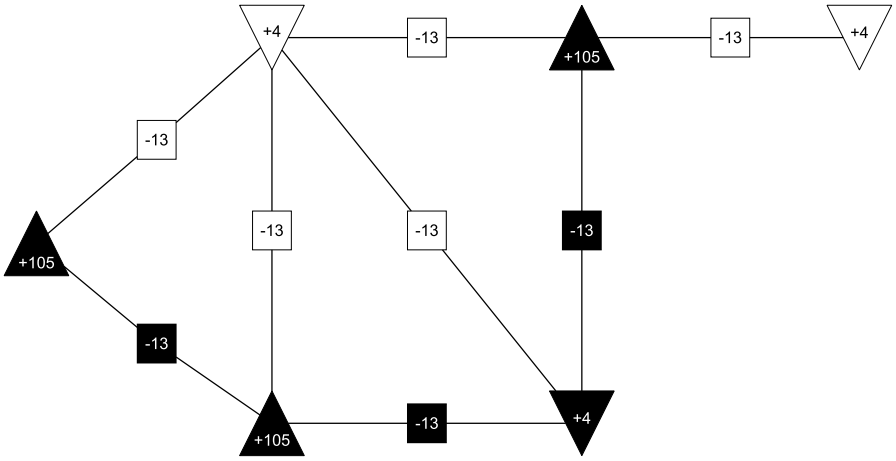


Figure 33 – Representation of a graph G built by the reduction from Steiner Tree.

C_{2q} and $\triangle$, a C_{2r}. Triangle edges represent edges that are incident to distinct vertices in its relative subgraph. Inside each of these symbols, we show the weight of a perfect matching in the corresponding subgraph.

Note that the solution contains edges from C_{210} cycles, as well as at most three path subgraphs P_{26}. Hence, there are three Steiner Tree solutions, $G[\{a,c,d,e\}]$, $(G[\{a,b,c,d\}] - ab)$ and $(G[\{a,b,c,d\}] - bc)$.

3.2.4 Polynomial-time algorithms

So far, we have proven that Weighted Connected Matching is NP-complete even when some constraints are imposed, such as limits on the weights, planarity and degree bounds. In this section, we turn our attention to other tractable cases, presenting algorithms for graphs of maximum degree two, trees and, more generally, graphs of bounded treewidth.

3.2.4.1 Graphs having degree at most 2

Due to Theorem 25, we know that Maximum Weight Connected Matching is NP-hard for graphs of maximum degree three. As such, we turn our attention to graphs of maximum degree two, i.e. the disjoint unions of paths and cycles, and prove that they allow for a linear time algorithm; in fact, for the case of paths, we reuse the algorithm for trees previously described.

As for cycles, we use the following method. Given graph $G = (V,E)$, take two arbitrary edges $E' = \{uw, wv\} \subseteq E$. A maximum connected matching in G contains one or no elements of E'. So, we compare three maximum connected matchings, defined in $G - uw$, $G - wv$, and $G - w$; they can be obtained in linear time, by using a dynamic programming if $|E' \cap M| = 1$, or by our linear algorithm for trees (see Section 3.2.4.3) otherwise. Among these three, the matching having the largest weight is a maximum weight connected matching.

Theorem 26. Maximum Weight Connected Matching can be solved in linear time for cycles. Furthermore, the problem can be solved in linear time for graphs of maximum degree at most 2.

Proof. Let $e_1, \ldots, e_n$ be the edges of the cycle C. First, we will compute the maximum weight connected matching containing e_n. Let $s_i = \sum_{j=1}^{i} w(e_{n-2j})$ for $i < \lfloor (n-1)/2 \rfloor$ and let $s'_i = \max_{1 \leq j \leq i} s_j$. Similarly, let $p_i = \sum_{j=1}^{i} w(e_{2j})$ for $i < \lfloor (n-1)/2 \rfloor$. Finally, let $s_0 = p_0 = 0$. Note that a matching containing edge e_n can be written as $\{e_2, \ldots, e_{2i}\} \cup \{e_n, \ldots, e_{n-2j}\}$, for some $i + j \leq n/2 - 1$, and its weight is given by $p_i + w(e_n) + s_j$. Also, note that, for fixed i, the maximum matching among every possible j is given by $S_i = p_i + w(e_n) + s'_j$, for $j = \lfloor n/2 \rfloor - i - 1$. Since all s_i, s'_i and p_i can be precomputed in linear time, we can compute each S_i in constant time, which allows us to compute the maximum weight connected matching containing e_n in linear time.

Since e_n was chosen arbitrarily, we can do the same procedure for a different edge, e_{n-1}. If neither e_{n-1} nor e_n belongs to a maximum weight connected matching, we know that $\{v\} = e_{n-1} \cap e_n$ is not saturated by it. Hence, we can delete v from C and compute the maximum weight connected matching of the resulting graph using the algorithm for trees from Section 3.2.4.3.

Since any maximum weight connected matching contains either e_{n-1}, e_n or none of them, if we take the maximum over these three cases, we have the maximum weight connected matching. □

3.2.4.2 Bounded treewidth graphs

This next algorithm, relies on the rank based approach of Bodlaender et al. [19] for optimizing dynamic programming algorithms for connectivity problems.

For each node x of a tree decomposition, our algorithm constructs a table $f_x(S,U) \subseteq \Pi(B_x) \times \mathbb{R}$, with $S,U \subseteq B_x$ and $\Pi(B_x)$ being the set of all partitions of B_x. Intuitively, each entry $(p,w) \in f_x(S,U)$ corresponds to a matching M of the subgraph induced by the bags of the subtree rooted at x with weight w, where each block $p \in \Pi(S \cup U)$ is part of a distinct connected component of $G[M]$.

Let U be a finite set, $\Pi(U)$ denote the set of all partitions of U, and $\sqsubseteq$ be the coarsening relation defined on $\Pi(U)$, i.e. given two partitions $p,q \in \Pi(U)$, $p \sqsubseteq q$ if and only if each block of q is contained in some block of p. It is known that $\Pi(U)$ together with $\sqsubseteq$ form a lattice, upon which we can define the usual *join* operator $\sqcup$ and *meet* operator $\sqcap$ [19]. The join operation $p \sqcup q$ works as follows: let H be the graph where $V(H) = U$ and $E(H) = \{uv \mid \{\{u,v\}\} \sqsubseteq p \vee \{\{u,v\}\} \sqsubseteq q\}$; $S \subseteq U$ is block of $p \sqcup q$ if and only if S induces a maximal connected component of H. The result of the meet operation $p \sqcap q$ is the unique partition such that each block is formed by the non-empty intersection between a block of p and a block of q. Given a subset $X \subseteq U$ and $p \in \Pi(U)$, $p_{\downarrow X} \in \Pi(X)$ is the partition obtained by removing all elements of $U \setminus X$ from p, while, for $Y \supseteq U$, $p_{\uparrow Y} \in \Pi(Y)$ is the partition obtained by adding to p a singleton block for each element in $Y \setminus U$. For $X \subseteq U$, we shorthand by $U[X]$ the partition where one block is precisely $\{X\}$ and all other are the singletons of $U \setminus X$; if $X = \{a,b\}$, we use $U[ab]$.

A set of *weighted partitions* of a ground set U is defined as $\mathcal{A} \subseteq \Pi(U) \times \mathbb{N}$. To speed up dynamic programming algorithms for connectivity problems, the idea is to only store a subset $\mathcal{A}' \subseteq \mathcal{A}$ that preserves the existence of at least one optimal solution. Formally, for each possible extension $q \in \Pi(U)$ of the current partitions of $\mathcal{A}$ to a valid solution, the optimum of $\mathcal{A}$ relative to q is denoted by $\mathsf{opt}(q, \mathcal{A}) = \min\{w \mid (p,w) \in \mathcal{A}, p \sqcup q = \{U\}\}$. $\mathcal{A}'$ *represents* $\mathcal{A}$ if $\mathsf{opt}(q, \mathcal{A}') = \mathsf{opt}(q, \mathcal{A})$ for all $q \in \Pi(U)$. The key result of [19] is given by Theorem 27.

Theorem 27 (3.7 of [19])**.** There exists an algorithm that, given $\mathcal{A}$ and U, computes $\mathcal{A}'$ in time $|\mathcal{A}|2^{(\omega-1)|U|}|U|^{\mathcal{O}}(1)$ and $|\mathcal{A}'| \leq 2^{|U|-1}$, where ω is the matrix multiplication constant.

A function $f : 2^{\Pi(U)\times\mathbb{N}} \times Z \mapsto 2^{\Pi(U)\times\mathbb{N}}$ is said to *preserve representation* if $f(\mathcal{A}', z) = f(\mathcal{A}, z)$ for every $\mathcal{A}, \mathcal{A}' \in \Pi(U) \times \mathbb{N}$ and $z \in Z$; thus, if one can describe a dynamic programming algorithm that uses only transition functions that preserve representation, it is possible to obtain $\mathcal{A}'$. In the following lemma, let $\mathsf{rmc}(\mathcal{A}) = \{(p,w) \in \mathcal{A} \mid \nexists (p, w') \in \mathcal{A}, w' < w\}$.

Lemma 29 (Proposition 3.3 and Lemma 3.6 of [19])**.** Let U be a finite set and $\mathcal{A} \subseteq \Pi(U) \times \mathbb{N}$. The following functions preserve representation and can be computed in $|\mathcal{A}| \cdot |\mathcal{B}| \cdot |U|^{\mathcal{O}}(1)$ time.

- **Union.** For $\mathcal{B} \in \Pi(U) \times \mathbb{N}$, $\mathcal{A} \uplus \mathcal{B} = \mathsf{rmc}(\mathcal{A} \cup \mathcal{B})$.
- **Insert.** For $X \cap U = \emptyset$, $\mathsf{ins}(X, \mathcal{A}) = \{(p_{\uparrow X\cup U}, w) \mid (p,w) \in \mathcal{A}\}$.
- **Shift.** For any integer w', $\mathsf{shift}(w', \mathcal{A}) = \{(p,w + w') \mid (p,w) \in \mathcal{A}\}$.
- **Glue.** Let $\hat{U} = U \cup X$, then $\mathsf{glue}(X, \mathcal{A}) = \mathsf{rmc}\left(\left\{(\hat{U}[X] \sqcup p_{\uparrow\hat{U}}, w) \mid (p,w) \in \mathcal{A}\right\}\right)$.
- **Project.** $\mathsf{proj}(X, \mathcal{A}) = \mathsf{rmc}\left(\{(p_{\downarrow\overline{X}}, w) \mid (p,w) \in \mathcal{A}, \forall u \in X : \exists v \in \overline{X} : p \sqsubseteq U[uv]\}\right)$, if $X \subseteq U$.
- **Join.** If $\hat{U} = U \cup U'$, $\mathcal{A} \subseteq \Pi(U) \times \mathbb{N}$ and $\mathcal{B} \in \Pi(U') \times \mathbb{N}$, then $\mathsf{join}(\mathcal{A}, \mathcal{B}) = \mathsf{rmc}\left(\{(p_{\uparrow\hat{U}} \sqcup q_{\uparrow\hat{U}}, w + w') \mid (p,w) \in \mathcal{A}, (q, w') \in \mathcal{B}\}\right)$.

Theorem 28. Given a tree decomposition of width t of the n-vertex input graph, Maximum Weight Connected Matching can be solved in $2^{\mathcal{O}}(t)\, n^{\mathcal{O}}(1)$ time.

Proof. Let G be the input graph to Maximum Weight Connected Matching, $\rho : E(G) \mapsto \mathbb{R}$ be the weighting of the edges, and $\mathbb{T}^* = (T^*, \mathcal{B}^*)$ be a tree decomposition of width t of G; in a slight abuse of notation, for $L \subseteq E(G)$, we define $\rho(L) = \sum_{e\in L} \rho(e)$. Without loss of generality, we suppose $\mathbb{T}^*$ is a nice tree decomposition, otherwise it can be found in polynomial time. We also assume that G is connected since, if it is not connected, we can run an algorithm in parallel for each component, which have treewidth bounded by that of G, and output the maximum of all these distinct components. In the first step of our algorithm, we pick a vertex $\pi \in V(G)$ and create a tree decomposition $\mathbb{T} = (T, \mathcal{B})$ rooted at a node r that corresponds to an empty bag that is also a forget bag for vertex π; for each such choice of π we run the dynamic programming algorithm we describe in the remainder of this proof. For each node $x \in V(T)$, we compute the table $f_x(S,U) \subseteq \Pi(B_x) \times \mathbb{R}$, with $S \subseteq B_x$ and $U \subseteq B_x \setminus S$. If we have a weighted partition $(p, w) \in f_x(S,U)$, then we want to ensure that there is a (partial) solution M_x with the following properties: (i) every vertex of S is already matched in G_x, (ii) vertices of U are half-matched, i.e. they have yet to be matched to other vertices but are used to determine connectivity of $G[M_x]$, and (iii) $\rho(M_x) = \sum_{e\in M_x} \rho(e) = w$. Note that vertices in U must already be accounted for when determining the connected components induced by M_x. After every operation involving families of partitions, we apply the algorithm of Theorem 27. We divide our analysis into the four cases of the definition of a nice tree

decomposition, where x is the bag for which we currently want to compute the dynamic programming table.

Leaf node. Since $B_x = \emptyset$, the only possible matching of G_x is the empty matching, so we define:

$$f_x(\emptyset, \emptyset) = \{(\emptyset, 0)\}$$

Introduce node. Let y be the child of of x and $\{v\} = B_x \setminus B_y$. We define our transition as follows, where $\mathcal{A}_x(S, U, u) = \mathsf{ins}(\{v\}, f_y(S \setminus \{u,v\}, U \cup \{u\}))$:

$$f_x(S,U) = \begin{cases} f_y(S,U), & \text{if } v \notin S \cup U; \\ \mathsf{glue}(N_S(u), \mathsf{ins}(\{v\}, f_y(S, U \setminus \{v\})), & \text{if } v \in U; \\ \biguplus_{u \in N_S(v)} \mathsf{shift}(\rho(uv), \mathsf{glue}\,(N_S(v), \mathcal{A}_x(S,U,u))), & \text{otherwise.} \end{cases}$$

If v is not in M_x, then M_x is also a partial solution to G_y which, by induction, is represented by an element of $f_y(S,U)$ and is covered by the first case of the equation. On the other hand, if $v \in M_x$, then v is either matched to a vertex in S, or it is a half-matched vertex. If it is half matched, then $M_y = M_x \setminus \{v\}$ must be a partial solution to G_y and, furthermore, must be represented in $f_y(S, U \setminus \{v\})$, since its matched edges are the same as in M_x and all other half-matched vertices of M_x also exist in M_y; this situation is covered in the second case of the equation, where we must further coarsen the partition that represents M_y by joining the blocks that have neighbors of v in them. Finally, if $v \in S$ and $uv \in M_x$, then it must be the case that $u \in S$, since $v \notin V(G_y)$. As such, $M_y = \{u\} \cup M_x \setminus \{uv\}$ must be a partial solution to G_y with u being a half-matched vertex, i.e. M_y must be represented in $f_y(S \setminus \{u,v\}, U \cup \{u\})$, which holds by induction. This final case is represented in the third case of the previous equation; note that we must add the weight of uv to the weight of M_y and join its connected components that contain vertices of $N_S(v)$.

Forget node. Let y be the child of x and $\{v\} = B_y \setminus B_x$. We compute our table as follows:

$$f_x(S,U) = f_y(S,U) \uplus \mathsf{proj}(\{v\}, f_y(S \cup \{v\}, U))$$

First, consider the case where $v \notin M_x$, and note that M_x is also a partial solution to G_y and, consequently, must be represented by $f_y(S,U)$. On the other hand, if $v \in M_x$, then v must be matched to some vertex of M_x, otherwise M_x would not be extensible to a matching (i.e. without half-matched vertices) of G, since $v \notin G \setminus G_x$. These two cases are represented by the two right-hand-side terms of our previous equation.

Join node. Finally, if y,z are the children of x, then we compose our table according to the following equation:

$$f_x(S,U) = \biguplus_{Y \subseteq S} \mathsf{join}(f_y(Y, U \cup (S \setminus Y)), f_z(S \setminus Y, U \cup Y))$$

Where the union operator runs over all subsets of S. Let M_x be a partial solution of G_x, $Y \subseteq S$ be the vertices in S matched to vertices of G_y, and M_y be the subset of M_x restricted to G_y. Observe that $M_y \cap S = Y$ since the vertices of Y are precisely those of B_y matched in M_y. Moreover, $M_y \cap B_y \setminus Y = U \cup (S \setminus Y)$ are the vertices of M_x not matched in M_y; they must, however, be half-matched vertices of M_y since

they are (half-)matched vertices of M_x. Consequently, M_y is represented by a partition $(p, w_y) \in f_y(Y, U \cup (S \setminus Y))$. Now, let $M_z \subseteq M_x$ be the partial solution to G_z where $S \setminus Y$ are the vertices of S not matched by M_y. Note that $U \cup Y$ are precisely the half-matched vertices of M_z since they must be either half-matched in M_x (U) or matched in M_y. As such, M_z is represented by a partition $(q, w_z) \in f_z(S \setminus Y, U \cup Y)$. Finally, we have that $p \sqcup q$ yields the same partition of $S \cup U$ as M_x since $M_y \cup M_z = M_x$, and, furthermore, $\rho(M_x) = \rho(M_y) + \rho(M_z) = w_y + w_z$ since no edge matched by M_y is present in M_z and vice-versa and $E(M_y) \cup E(M_z) = E(M_x)$. These are the exact properties given by the join operation; since the above equation runs over all subsets of S, M_x will be represented by $f_x(S, U)$.

In order to obtain the solution to G, first observe that, since G is connected and the root r of T is a forget bag for π, G has a connected matching of weight w if and only if $\{(\{\pi\}, w)\}$ is the unique element of $f_{r'}(\{\pi\}, \emptyset)$, where r' is the child bag of r. In the final step of the algorithm, we return the maximum weight obtained between all choices of π.

As to the running time of the dynamic programming algorithm, note that, for each choice of π we can compute all entries of f_x in time bounded by $2^{|B_x|} \cdot \sum_{i=0}^{|B_x|} \binom{|B_x|}{i} 2^{\omega i} t^{\mathcal{O}}(1) \leq (1 + 2^{\omega})^{2t} t^{\mathcal{O}}(1)$; the term $2^{\omega i}$ corresponds to the time needed to execute the algorithm of Theorem 27 for an entry where $|S| = i$ and the term $2^{|B_x|}$ comes from all possible choices of U. Forget nodes can be computed in the same time since we make the same number $\mathcal{O}(t)$ fewer calls to Theorem 27 for each entry. Finally, leaves can be solved in constant time and table f_x for a join node x can be computed in $2^{|B_x|} \cdot \sum_{i=0}^{|B_x|} \binom{|B_x|}{i} 2^{\omega i + i} t^{\mathcal{O}}(1) \leq (1 + 2^{2\omega})^{2t} t^{\mathcal{O}}(1)$ time; in this case, the $2^{\omega i + i}$ terms comes from the 2^i choices for Y, each of which requires one invocation of Theorem 27. Given that we have $\mathcal{O}(nt)$ nodes in a nice tree decomposition, our dynamic programming algorithm can be computed in $\mathcal{O}(nt)$ times the cost of the most expensive nodes, which are the join nodes, totaling the required $2^{\mathcal{O}}(t)\, n^{\mathcal{O}}(1)$ time. Since we have to apply it for each $\pi \in V(G)$, the entire algorithm runs in $2^{\mathcal{O}}(1)\, n^{\mathcal{O}}(1)$ time. □

3.2.4.3 Trees

The algorithm described in Section 3.2.4.2 implies that Weighted Connected Matchings can be solved in $n^{\mathcal{O}(1)}$ time on trees. We strengthen this result with a linear time algorithm for this class.

Given a tree T, we begin by rooting it in some vertex $r \in V(T)$. Then, we traverse this rooted tree in post-order such a way that, when visiting vertex v, we find the weight of a maximum weight connected matching in the subtree defined by v and its descendants and such that v is saturated. The matching having the largest weight is a maximum weight connected matching in T.

In Theorem 29, we give details about this algorithm, proving its correctness and analyzing the running time.

Theorem 29. Maximum Weight Connected Matching on trees can be solved in linear time.

Proof. We describe a linear algorithm that solves Weighted Connected Matching for trees. Let $T = (V,E)$ be a tree. We denote by T^r the tree T rooted in r. Moreover, for $v \in V$, T^r_v is the subtree of T^r that contains v and all its descendants, and $S(r,v)$ is the set of children of v in T^r. Also, consider $B_{r,v}$ as the weight of a maximum weight connected matching in T^r_v such that, if v is not a leaf, then v is saturated. Moreover, $\overline{B}_{r,v}$

is the weight of a matching M defined as the union of the maximum connected matchings in T_u^r, for each $u \in S(r,v)$, such that $G[V(M) \cup \{v\}]$ is connected.

Next, we describe a dynamic programming algorithm that, for given a root $r \in V$, can be used to obtain $\overline{B}_{r,v}$ and $B_{r,v}$ for every $v \in V$. For the base case, the vertex v is a leaf in T_v^r, and then $B_{r,v} = \overline{B}_{r,v} = 0$. Otherwise, v is not a leaf, and we can obtain $B_{r,v}$, $\overline{B}_{r,v}$ as follows.

$$f(r,vu) = \overline{B}_{r,u} + w(vu) + \sum_{s \in S(r,v) \setminus \{u\}} \max\{B_{r,s},0\}$$

$$\overline{B}_{r,v} = \sum_{u \in S(r,v)} \max\{B_{r,u},0\}$$

$$B_{r,v} = \max_{u \in S(r,v)} f(r,vu)$$

Next, we show that we can run this dynamic programming algorithm in linear time. Clearly, summations $\overline{B}_{r,v} = \sum_{u \in S(r,v)} \max\{B_{r,u},0\}$ for all vertices $v \in V$ can be calculated in linear time. For $B_{r,v}$, note that it can also be written as follows.

$$B_{r,v} = \overline{B}_{r,v} + \max_{u \in S(r,v)} (w(uv) + \overline{B}_{r,u} - B_{r,u})$$

This leads to a linear time procedure to obtain $B_{r,v}$ and $\overline{B}_{r,v}$, for every $v \in V$. Now, we can find a maximum weight connected matching in T by reconstructing the matching that generated weight $B_{r,h}$, for the vertex h that maximizes $B_{r,h}$. □

3.2.4.4 Example of the tree algorithm

As an example, consider as input the tree illustrated in Figure 34. We define $b_{r,v}$ and $M(T_v^r)$, which can be used to reconstruct our dynamic programming algorithm, as follows.

$$b_{r,v} = \arg\max_{u \in S(r,v)} f(r,vu)$$

$$M(T_v^r) = \begin{cases} \{\}, & \text{if } B_{r,v} \leq 0 \\ \{vb_{r,v}\} \cup \left(\bigcup_{u \in S(r,v) \setminus \{b_{r,v}\}} M(T_u^r) \right) \cup \left(\bigcup_{u \in S(r,b_{r,v})} M(T_u^r) \right), & \text{otherwise} \end{cases}$$

The vertex a is chosen to be the root. In Table 8, we show the related variables obtained. Also, the rows are ascending in the same order as those variables can be calculated using our dynamic programming algorithm. In this example, the vertex h that maximizes $B_{a,h}$ is a. So, we build the matching $M(T_a^a)$, which is defined in T_a^a. First, we saturate a with $b_{a,a} = b$. Then we add the following partial matchings.

$$M(T_e^a) = \{eb_{a,e}\} = \{ej\}$$
$$M(T_c^a) = \{cb_{a,c}\} = \{cf\}$$
$$M(T_d^a) = \{\}$$

Note that, though the subtree T_d^a is not empty, there is no possible matching that can be added to increase the weight of a connected matching containing a. Then, the subtree T_d^a is discarded, and we set $M(T_d^a) = \{\}$.

Finally, we obtain the maximum weight connected matching $\{ab,ej,cf\}$, illustrated in Figure 34, whose weight is $B_{a,a} = 12$.

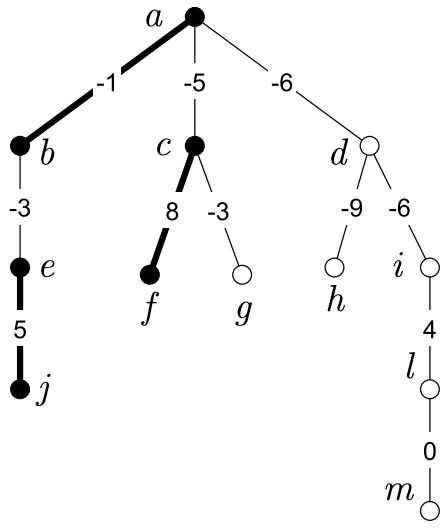


Figure 34 – A tree and a maximum weight connected matching.

v	$B_{r,v}$	$\overline{B}_{r,v}$	$b_{r,v}$
j	0	0	–
e	5	0	j
b	-3	5	e
f	0	0	–
g	0	0	–
c	8	0	f
h	0	0	–
m	0	0	–
l	0	0	m
i	4	0	l
d	-5	4	h
a	12	8	b

Table 8 – Values of $B_{r,v}$, $\overline{B}_{r,v}$, and $b_{r,v}$ for each vertex v.

3.2.5 Kernelization

Theorem 28 implies that Maximum Weight Connected Matching parameterized by treewidth is in FPT, which immediately prompts an investigation into whether its decision version admits a polynomial kernel under the same parameterization. We answer this negatively by showing that Weighted Connected Matching parameterized by vertex cover number does not admit a polynomial kernel, unless NP $\subseteq$ coNP/poly, even if the input is restricted to bipartite graphs of bounded diameter and the allowed weights are in $\{0, 1\}$, which implies the same result when parameterizing by treewidth, since treewidth is upper bounded by the vertex cover number.

We prove this result through an OR-cross-composition [20] from the 3SAT problem. The construction is heavily inspired by the proof described in Section 3.2.2. Formally let, $\mathcal{H} = \{(X_1, \mathcal{C}_1), \ldots, (X_t, \mathcal{C}_t)\}$ be a set of t 3SAT instances such that $X_i = X = \{x_1, \ldots, x_n\}$ for every $i \in [t]$. Also, let $\mathcal{C} = \bigcup_{i \in [t]} \mathcal{C}_i$. Finally, let (G, k) be the Weighted Connected Matching instance we are going to build.

We begin the construction by adding to G a pair of vertices c_j,c'_j for each $C_j \in \mathcal{C}$ and a unit weight edge between them. Then, for each $x_i \in X$, we add vertices x_i^-, x_i^*, x_i^+ and edges $x_i^- x_i^*, x_i^* x_i^+$, each of weight 1. Now, for each $C_j \in \mathcal{C}$ and $i \in [n]$, if $x_i \in C_j$, we add the 0-weight edge $x_i^+ c_j$ to G, otherwise, if $\overline{x_i} \in C_j$, we add the weight 0 edge $x_i^- c_j$. We conclude this first part of the construction by adding a pair of vertices h,h' to G, making them adjacent with an edge of weight 1, and adding an edge of weight 0 between h and x_i^* for every $x_i \in X$. At this point, we have an extremely similar graph to the one constructed in Section 3.2.2.

For the next part of the construction, we add a copy of $K_{1,t}$, where the vertex on the smaller side is labeled q and, the vertices on the other side are each assigned a unique label from the set $Y = \{y_1, \ldots, y_t\}$, with each edge having weight 1. Now, for each $y_\ell \in Y$ and $C_j \in \mathcal{C} \setminus \mathcal{C}_\ell$, we add the 0-weight edges $c'_j y_\ell$ and $h y_\ell$. Finally, we set $k = |\mathcal{C}| + |X| + 2$, i.e.

we must pick one edge in each clause gadget and vertex gadget plus the edge hh' and one edge between q and Y. Note that $|V(G)| = 3|X| + 2|\mathcal{C}| + |Y| + 3 \leq 3|X| + 2|X|^3 + |Y| + 3$, which implies that $V(G) \setminus Y$ is a vertex cover of G of size $\mathcal{O}(|X|^3)$, as required by the cross-composition framework. Moreover, note that G is bipartite, as we can partition it as follows: $L = \{q, h\} \cup \{x_i^+, x_i^- \mid i \in [n]\} \cup \{c'_j \mid C_j \in \mathcal{C}\}$ and $R = V(G) \setminus L$, where both L and R are independent sets.

Lemma 30. If $(X, \mathcal{C}_\ell)$ admits a solution, then (G, k) also admits a solution.

Proof. Let φ be a satisfying assignment for $(X, \mathcal{C}_\ell)$. We build the solution M for (G, k) as follows. First, for each $x_i \in X$, if $\varphi(x_i) = 0$, add edge $x_i^- x_i^*$ to M, otherwise add edge $x_i^+ x_i^*$ to M, with a total weight of $|X|$ after this step. Now, for each $C_j \in \mathcal{C}$, and $c_j c'_j$ to M, reaching $|\mathcal{C}| + |\mathcal{C}|$ weight. Finally, add hh' and qy_ℓ to M, so now M has $|X| + |\mathcal{C}| + 2 = k$ weight. Note that M is a matching. To see that it induces a connected graph, first observe that $\{q, y_\ell, h, h', x_1^*, \ldots, x_n^*\} \cup \bigcup_{i \in [n]} \{x_i^-, x_i^+\} \cap M$ are all part of the same connected component Q. Moreover, for every $C_j \notin \mathcal{C}_\ell$, we have both c'_j and c_j also belong to Q since $c'_j y_\ell \in E(G)$. For each $C_j \in \mathcal{C}_\ell$, suppose that $x_i \in C_j$ and $\varphi(x_i) = 1$; note that $x_i^+ \in Q$, so it holds that c_j and c'_j are also in Q, completing the proof. □

Lemma 31. If (G,k) admits a connected matching M of weight at least k, then there is some $(X, \mathcal{C}_\ell) \in \mathcal{H}$ that admits a solution.

Proof. First, note that k is also the weight of a maximum weighted matching of G, which is achieved by picking all edges $c_j c'_j$, edge hh', one edge incident to q, and one edge of weight one incident to each x_i^*. As such, we observe that there is one edge $qy_\ell \in M$ and, furthermore, no other $y_p \in M$, otherwise they would either be matched to h or to some c'_j; in either case we would have $w(M) < k$, since we would be replacing an edge of weight one with one of weight zero, and neither h' nor c_j can be matched with other edges of larger weight. Moreover, this implies that each x_i^* is matched to either x_i^+ or x_i^-, otherwise we would also not be able to achieve the necessary weight. As such, for each x_i, we set $\varphi(x_i) = 1$ if and only if x_i^+ is matched to x_i^*. Finally, note that, for each $C_j \in C_\ell$, there must be a path between c'_j and q passing through some x_i^* and, furthermore, this path must pass through either x_i^+ if $x_i \in C_j$ or x_i^- if $\overline{x_i} \in C_j$. This, in turn, implies that there is a literal of C_j that evaluates to true and satisfies C_j. Consequently, every C_j is satisfied and φ is a solution to $(X, \mathcal{C}_\ell)$. □

Combining the two previous lemmas, we immediately obtain the next theorem.

Theorem 30. Unless NP ⊆ coNP/poly, Weighted Connected Matching does not admit a polynomial kernel when parameterized by vertex cover number and required weight even if the input graph is bipartite and edge weights are in $\{0, 1\}$.

CONCLUSIONS AND FUTURE WORK

In this thesis, we have studied the complexity of problems regarding two $\mathscr{P}$-matchings: connected matchings and disconnected matchings.

We have presented c-disconnected matchings and the corresponding decision problem, which we named Disconnected Matching. This problem is related to the well-studied induced matchings and the problem of recognizing graphs that admit a sufficiently large induced matching. Our results show that, when the number of connected components c is fixed, c-Disconnected Matching is solvable in polynomial time if $c = 1$ but NP-complete even on bipartite graphs if $c \geq 2$. We also proved that unlike Induced Matching, Disconnected Matching remains NP-complete on chordal graphs and bounded degree graphs. Besides, we proved that it is NP-hard to decide if the equality $\beta_{d,i}(G) = \beta_{d,j}(G)$ holds for a given graph G and $1 \leq i < j$.

On the positive side, we show that the problem can be solved in polynomial time for interval graphs, in XP time for graphs with a polynomial number of minimal separators when parameterized by the number of connected components c, and in FPT time when parameterized by treewidth. Finally, we showed that Disconnected Matching does not admit polynomial kernels for very powerful parameters, namely vertex cover and vertex deletion distance to clique, by showing that this holds for the Induced Matching particular case.

Concerning connected matchings, we presented an algorithm that finds a maximum connected matching in any graph in linear time, given a maximum matching, whose problem we named Maximum Connected Matching. Besides, motivated by previous works on weighted $\mathscr{P}$-matchings, such as Weighted Induced Matching [115, 150] and Weighted Acyclic Matching [63], in this thesis we introduced and studied Weighted Connected Matching problem.

We began our investigation on the complexity of Weighted Connected Matching by imposing restrictions on the input graphs and weights. In particular, we showed that the problem is NP-complete on planar bipartite graphs and bipartite graphs of diameter 4 for binary weights, and on subcubic planar graphs and starlike graphs when weights are restricted to $\{-1, +1\}$. On the positive side, we presented polynomial-time algorithms for Maximum Weight Connected Matching on chordal graphs with non-negative weights, graphs having maximum degree at most two with arbitrary weights, on trees and, more generally, on graphs of bounded treewidth. The latter algorithm implies that Weighted Connected Matching is fixed-parameter tractable under the treewidth parameterization. This prompted our study of the problem from the kernelization point of view; our inquiry showed that no polynomial kernel exists when parameterized by vertex cover and the minimum required weight unless NP $\subseteq$ coNP/poly.

Future work

Possible directions for future work include studies related to $\mathscr{P}$-matchings. In particular, there are many open problems related to connected matchings and disconnected matchings.

Although we presented the complexity of Disconnected Matching for bipartite graphs, interval graphs, and chordal graphs, we could approach these problems on other graph classes. In particular, we would like to know the complexity of Disconnected Matching for strongly chordal graphs; we note that the reduction presented in Section 2.2 has many induced subgraphs isomorphic to a sun graph.

Another interesting goal would be to find graph classes with an exponential number of minimal separators for which Disconnected Matching could be solved in polynomial time. One example is k-thetas[1], where we could easily obtain an induced matching of size k, which is also a k-disconnected matching. However, the solution is simple and the graph class is too strict. So, we are interested in non-trivial graph classes and solutions for this problem.

Also, we would like to know the complexity of obtaining a minimum maximal disconnected matching in a given graph. Besides, we are interested in the characterizations of graphs with the size of a minimum maximal disconnected matching (β_d^-) equals the matching number or the disconnected matching number. Three simple examples that contrast these numbers are: (i) a k-theta, where $\beta_d^- = 2$, and $\beta = \beta_d = k$, (ii) a C_6, where $\beta_d^- = \beta_d = 2 < 3 = \beta$, and (iii) a P_5, where $\beta_d^- = \beta_d = \beta = 2$.

On its weighted version, we would like to know if there are graph classes or weight constraints for which Weighted Disconnected Matching could be solved in polynomial time. In this problem, we are given the input (G,k) and we want to know if the edge-weighted graph G admits a disconnected matching whose sum of its edge weights is at least k.

We are also interested in the parameterized complexity of the problem. Our results show that, when parameterized by c, the problem is paraNP-hard; on the other hand, when parameterized by the number of edges in the matching, it is W[1]-hard since Induced Matching is W[1]-hard under this parameterization [143]. A first question of interest is whether chordal graphs admit an FPT algorithm when parameterized by c; while the algorithm presented in Section 2.3.1 works for all classes with a polynomial number of minimal separators, chordal graphs offer additional properties that may aid in the proof of an FPT algorithm.

Another research direction would be the investigation of other structural parameterizations, such as vertex cover and cliquewidth; while the former yields a fixed-parameter tractable algorithm due to Theorem 15, we would like to know if we can find a single exponential-time algorithm under this weaker parameterization. On the other hand, cliquewidth is a natural next step, as graphs of bounded treewidth have bounded cliquewidth, but the converse does not hold. Finally, while we have settled several kernelization questions for Disconnected Matching and Induced Matching, other parameterizations are still of interest, such as max leaf number, feedback edge set, and neighborhood diversity.

Concerning connected matchings, we have strong results on its unweighted version, since we can obtain maximum connected matchings as fast as maximum matchings.

[1]A k-theta is a graph G with vertex set $V(G) = \{a, a_1, \ldots, a_k, b, b_1, \ldots, b_k\}$, and its set of edges is $\{aa_i, bb_i, a_ib_i \mid 1 \leq i \leq k\}$

However, minimum maximal connected matchings have not been studied yet. So, we want to know the complexity of obtaining a maximal connected matching of size at most k in a given graph. Note that, unlike the equality $\beta = \beta_c$ holds, we would like to know if these parameters, or the size of a minimum maximal matching (β^-), are comparable to the size of a minimum maximal connected matching (β_c^-). Note that β_c^- can be larger than β^- or not; for example, in a C_6, $\beta^- = 2 > 3 = \beta_c^- = \beta$, while, in a C_5, $\beta^- = \beta_c^- = \beta = 2$. So we would like to study the cases and characterizations where the equalities $\beta_c^- = \beta^-$ and $\beta_c^- = \beta$ hold.

Concerning its weighted version, Weighted Connected Matchings, we could approach it by different combinations of graph classes and allowed edge weights. Specifically, we would like to know the complexity of Weighted Connected Matching for diameter 3 bipartite graphs when weights are non-negative, chordal bipartite graphs, and subcubic planar graphs under the same constraint. Other graph classes of interest include cactus graphs and block graphs.

We are also interested in the parameterized complexity of the problem. In terms of natural parameterizations, we see two possible directions: parameterizing by the number of edges in the matching or by the weight of the matching; while we have some negative kernelization results for these parameters, tractability is still unknown. Other possibilities include the study of other structural parameterizations, with the main open question being tractability for the cliquewidth parameterization.

REFERENCES

[1] Abraham, D. J., Cechlárová, K., Manlove, D. F., and Mehlhorn, K. Pareto optimality in house allocation problems. In *Algorithms and Computation* (Berlin, Heidelberg, 2005), R. Fleischer and G. Trippen, Eds., Springer Berlin Heidelberg, pp. 3–15.

[2] Abraham, D. J., Irving, R. W., Kavitha, T., and Mehlhorn, K. Popular matchings. *SIAM Journal on Computing 37*, 4 (2007), 1030–1045.

[3] Abrishami, T., Chudnovsky, M., Dibek, C., Thomassé, S., Trotignon, N., and Vušković, K. Graphs with polynomially many minimal separators. *arXiv preprint arXiv:2005.05042* (2020).

[4] Al-Herz, A., and Pothen, A. A 2/3-approximation algorithm for vertex-weighted matching. *Discrete Applied Mathematics 308* (2022), 46–67. Combinatorial Optimization ISCO 2018.

[5] Albers, S., and Schubert, S. Optimal algorithms for online b-matching with variable vertex capacities. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021)* (Dagstuhl, Germany, 2021), M. Wootters and L. Sanità, Eds., vol. 207 of *Leibniz International Proceedings in Informatics (LIPIcs)*, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, pp. 2:1–2:18.

[6] Andrews, M., Atallah, M., Chen, D., and Lee, D. Parallel algorithms for maximum matching in interval graphs. In *Proceedings of 9th International Parallel Processing Symposium* (1995), pp. 84–92.

[7] Balakrishnan, H., Barrett, C. L., Kumar, V. A., Marathe, M. V., and Thite, S. The distance-2 matching problem and its relationship to the mac-layer capacity of ad hoc wireless networks. *IEEE Journal on Selected Areas in Communications 22*, 6 (2004), 1069–1079.

[8] Ball, M. O., and Derigs, U. An analysis of alternative strategies for implementing matching algorithms. *Networks 13*, 4 (1983), 517–549.

[9] Baste, J., Fürst, M., and Rautenbach, D. Acyclic matchings in graphs of bounded maximum degree. *Discrete Mathematics 345*, 7 (2022), 112885.

[10] Baste, J., and Rautenbach, D. Degenerate matchings and edge colorings. *Discrete Applied Mathematics 239* (2018), 38–44.

[11] Baste, J., Rautenbach, D., and Sau, I. Uniquely restricted matchings and edge colorings. In *Graph-Theoretic Concepts in Computer Science* (Cham, 2017), H. L. Bodlaender and G. J. Woeginger, Eds., Springer International Publishing, pp. 100–112.

[12] Bell, C. E. Weighted matching with vertex weights: An application to scheduling training sessions in nasa space shuttle cockpit simulators. *European Journal of Operational Research 73*, 3 (1994), 443–449.

[13] Berge, C. Two theorems in graph theory. *Proceedings of the National Academy of Sciences 43*, 9 (1957), 842–844.

[14] Berry, A., and Simonet, G. Computing a clique tree with algorithm MLS (maximal label search). *CoRR abs/1610.09623* (2016).

[15] Biasi, M. D. Max-weight connected subgraph problem in planar graphs. Theoretical Computer Science Stack Exchange, 2014.

[16] Bienkowski, M., Fuchssteiner, D., Marcinkowski, J., and Schmid, S. Online dynamic b-matching: With applications to reconfigurable datacenter networks. *SIGMETRICS Perform. Eval. Rev. 48*, 3 (mar 2021), 99–108.

[17] Bilò, V., Flammini, M., Monaco, G., and Moscardelli, L. On the impact of buyers preselection in pricing problems. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems* (2018), pp. 1871–1873.

[18] Blair, J., and Peyton, B. An introduction to chordal graphs and clique trees.

[19] Bodlaender, H. L., Cygan, M., Kratsch, S., and Nederlof, J. Deterministic single exponential time algorithms for connectivity problems parameterized by treewidth. *Information and Computation 243* (2015), 86 – 111. 40th International Colloquium on Automata, Languages and Programming (ICALP 2013).

[20] Bodlaender, H. L., Jansen, B. M. P., and Kratsch, S. Cross-composition: A new technique for kernelization lower bounds. In *Proc. of the 28th International Symposium on Theoretical Aspects of Computer Science (STACS)* (2011), vol. 9 of *LIPIcs*, pp. 165–176.

[21] Bondy, J. A., and Murty, U. S. R. *Graph Theory with Applications*. Elsevier, New York, 1976.

[22] Bondy, J. A., and Murty, U. S. R. *Graph Theory*. Springer, 2008.

[23] Brandstädt, A., and Hoàng, C. T. Maximum induced matchings for chordal graphs in linear time. *Algorithmica 52*, 4 (Dec 2008), 440–447.

[24] Brandstädt, A., Le, V. B., and Spinrad, J. P. *Graph Classes: A Survey*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1999.

[25] BrandstÄdt, A., Le, V., and Spinrad, J. *Graph Classes: A Survey*. Society for Industrial and Applied Mathematics, 1999.

[26] Brock, J. D. *Maximum Independent Set and Maximum Induced Matching Problems for Competitive Programming*. PhD thesis, Appalachian State University, 2021.

[27] Bruhn, H., and Joos, F. A stronger bound for the strong chromatic index. *Electronic Notes in Discrete Mathematics 49* (2015), 277–284. The Eight European Conference on Combinatorics, Graph Theory and Applications, EuroComb 2015.

[28] Cameron, K. Induced matchings. *Discrete Applied Mathematics 24*, 1 (1989), 97–102.

[29] Cameron, K. Induced matchings in intersection graphs. *Electronic Notes in Discrete Mathematics 5* (2000), 50–52.

[30] Cameron, K. *Connected Matchings.* Springer Berlin Heidelberg, Berlin, Heidelberg, 2003, pp. 34–38.

[31] Cameron, K., Sritharan, R., and Tang, Y. Finding a maximum induced matching in weakly chordal graphs. *Discrete Mathematics 266*, 1-3 (2003), 133–142.

[32] Chandran, L. S. A linear time algorithm for enumerating all the minimum and minimal separators of a chordal graph. In *Computing and Combinatorics* (Berlin, Heidelberg, 2001), J. Wang, Ed., Springer Berlin Heidelberg, pp. 308–317.

[33] Chang, J.-M. Induced matchings in asteroidal triple-free graphs. *Discrete Applied Mathematics 132*, 1-3 (2003), 67–78.

[34] Chang, M.-S. Algorithms for maximum matching and minimum fill-in on chordal bipartite graphs. In *Proceedings of the 7th International Symposium on Algorithms and Computation* (Berlin, Heidelberg, 1996), ISAAC '96, Springer-Verlag, p. 146–155.

[35] Chaudhary, J. *Matching and its variations: an algorithmic study.* PhD thesis, IIT Delhi, 2021.

[36] Chaudhary, J., and Panda, B. On the complexity of minimum maximal uniquely restricted matching. *Theoretical Computer Science 882* (2021), 15–28.

[37] Coleman, T. F., and Pothen, A. The null space problem ii. algorithms. *SIAM Journal on Algebraic Discrete Methods 8*, 4 (1987), 544–563.

[38] Cygan, M., Fomin, F. V., Kowalik, Ł., Lokshtanov, D., Marx, D., Pilipczuk, M., Pilipczuk, M., and Saurabh, S. *Parameterized algorithms*, vol. 3. Springer, 2015.

[39] Dagum, P., and Luby, M. Approximating the permanent of graphs with large factors. *Theoretical Computer Science 102*, 2 (1992), 283–305.

[40] Dahlhaus, E., and Karpinski, M. Matching and multidimensional matching in chordal and strongly chordal graphs. *Discrete Applied Mathematics 84*, 1-3 (May 1998), 79–91.

[41] Demange, M., and Ekim, T. Minimum maximal matching is np-hard in regular bipartite graphs. In *Theory and Applications of Models of Computation* (Berlin, Heidelberg, 2008), M. Agrawal, D. Du, Z. Duan, and A. Li, Eds., Springer Berlin Heidelberg, pp. 364–374.

[42] Demange, M., Ekim, T., and Tanasescu, C. Hardness and approximation of minimum maximal matchings. *International Journal of Computer Mathematics 91*, 8 (2014), 1635–1654.

[43] Deogun, J. S., Kloks, T., Kratsch, D., and Müller, H. On the vertex ranking problem for trapezoid, circular-arc and other graphs. *Discrete Applied Mathematics 98*, 1 (1999), 39–63.

[44] Dirac, G. A. On rigid circuit graphs. *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg 25*, 1 (Apr 1961), 71–76.

[45] Duan, R., Pettie, S., and Su, H.-H. Scaling algorithms for weighted matching in general graphs. *ACM Trans. Algorithms 14*, 1 (jan 2018).

[46] Duckworth, W., Manlove, D. F., and Zito, M. On the approximability of the maximum induced matching problem. *Journal of Discrete Algorithms 3*, 1 (2005), 79–91.

[47] Ducoffe, G., and Popa, A. The b-matching problem in distance-hereditary graphs and beyond. *Discrete Applied Mathematics 305* (2021), 233–246.

[48] Dudycz, S., Manurangsi, P., and Marcinkowski, J. Tight inapproximability of minimum maximal matching on bipartite graphs and related problems. In *Approximation and Online Algorithms* (Cham, 2021), J. Koenemann and B. Peis, Eds., Springer International Publishing, pp. 48–64.

[49] Dyer, M., and Muller, H. Counting perfect matchings and the switch chain. *SIAM Journal on Discrete Mathematics 33*, 3 (2019), 1146–1174.

[50] Edmonds, J. Maximum matching and a polyhedron with 0, 1-vertices. *Journal of Research of the National Bureau of Standards Section B Mathematics and Mathematical Physics 69B*, 1 and 2 (Jan. 1965), 125.

[51] Edmonds, J. Paths, trees, and flowers. *Canadian Journal of Mathematics 17* (1965), 449–467.

[52] Edmonds, J., and Johnson, E. L. Matching: a well-solved class of integer linear programs. In *in: Combinatorial structures and their applications (Gordon and Breach* (1970), pp. 89–92.

[53] Egerváry, E. Über kombinatorische eigenschaften von matrizen. *Mat. Fiz. Lapok 38* (1931), 16–28.

[54] Erdöus, P. Problems and results in combinatorial analysis and graph theory. In *Graph Theory and Applications*, J. Akiyama, Y. Egawa, and H. Enomoto, Eds., vol. 38 of *Annals of Discrete Mathematics*. Elsevier, 1988, pp. 81–92.

[55] Erveš, R., and Šparl, P. Maximum induced matching of hexagonal graphs. *Bulletin of the Malaysian Mathematical Sciences Society 39*, 1 (2016), 283–295.

[56] Escoffier, B., Monnot, J., Paschos, V. T., and Xiao, M. New results on polynomial inapproximabilityand fixed parameter approximability of edge dominating set. *Theory of Computing Systems 56*, 2 (June 2014), 330–346.

[57] Espelage, W., Gurski, F., and Wanke, E. How to solve np-hard graph problems on clique-width bounded graphs in polynomial time. In *International Workshop on Graph-Theoretic Concepts in Computer Science* (2001), Springer, pp. 117–128.

[58] Even, S., and Kariv, O. An $O(n^{2.5})$ algorithm for maximum matching in general graphs. In *16th Annual Symposium on Foundations of Computer Science (sfcs 1975)* (1975), pp. 100–112.

[59] Francis, M. C., Jacob, D., and Jana, S. Uniquely restricted matchings in interval graphs. *SIAM Journal on Discrete Mathematics 32*, 1 (2018), 148–172.

[60] Fredman, M. L., and Tarjan, R. E. Fibonacci heaps and their uses in improved network optimization algorithms. *J. ACM 34*, 3 (jul 1987), 596–615.

[61] Fürst, M. On the hardness of deciding the equality of the induced and the uniquely restricted matching number. *Information Processing Letters 147* (2019), 77–81.

[62] Fürst, M. *Restricted matchings.* PhD thesis, 2019.

[63] Fürst, M., and Rautenbach, D. On some hard and some tractable cases of the maximum acyclic matching problem. *Annals of Operations Research 279*, 1 (Aug 2019), 291–300.

[64] Gabow, H., Galil, Z., and Spencer, T. Efficient implementation of graph algorithms using contraction. In *25th Annual Symposium onFoundations of Computer Science, 1984.* (1984), pp. 347–357.

[65] Gabow, H. N. *Implementation of Algorithms for Maximum Matching on Nonbipartite Graphs.* PhD thesis, Stanford University, Stanford, CA, USA, 1974. AAI7413628.

[66] Gabow, H. N. An efficient implementation of edmonds' algorithm for maximum matching on graphs. *J. ACM 23*, 2 (apr 1976), 221–234.

[67] Gabow, H. N. An efficient reduction technique for degree-constrained subgraph and bidirected network flow problems. In *Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing* (New York, NY, USA, 1983), STOC '83, Association for Computing Machinery, p. 448–456.

[68] Gabow, H. N. The weighted matching approach to maximum cardinality matching. *Fundamenta Informaticae 154*, 1-4 (Aug. 2017), 109–130.

[69] Gabow, H. N. Data structures for weighted matching and extensions to b-matching and f-factors. *ACM Trans. Algorithms 14*, 3 (jun 2018).

[70] Gabow, H. N., and Sankowski, P. Algorithms for weighted matching generalizations ii: f-factors and the special case of shortest paths. *SIAM Journal on Computing 50*, 2 (2021), 555–601.

[71] Gabow, H. N., and Tarjan, R. E. Faster scaling algorithms for general graph matching problems. *J. ACM 38*, 4 (oct 1991), 815–853.

[72] Gale, D., and Shapley, L. S. College admissions and the stability of marriage. *The American Mathematical Monthly 69*, 1 (1962), 9–15.

[73] Galil, Z., Micali, S., and Gabow, H. An o(ev\logv) algorithm for finding a maximal weighted matching in general graphs. *SIAM Journal on Computing 15*, 1 (1986), 120–130.

[74] Gallai, T. Kritische graphen ii. *Magyar Tud. Akad. Mat. Kutato Int. Kozl. 8* (1963), 373–395.

[75] Gallai, T. Maximale systeme unabhangiger kanten. *Magyar Tud. Akad. Mat. Kutato Int. Kozl. 9* (1964), 401–413.

[76] Gardi, F. Efficient algorithms for disjoint matchings among intervals and related problems. In *Discrete Mathematics and Theoretical Computer Science* (Berlin, Heidelberg, 2003), C. S. Calude, M. J. Dinneen, and V. Vajnovszki, Eds., Springer Berlin Heidelberg, pp. 168–180.

[77] Garey, M. R., and Johnson, D. S. The rectilinear steiner tree problem is np-complete. *SIAM Journal on Applied Mathematics 32*, 4 (1977), 826–834.

[78] Garey, M. R., and Johnson, D. S. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979.

[79] Ghosal, P., Nasre, M., and Nimbhorkar, P. Rank-maximal matchings – structure and algorithms. *Theoretical Computer Science 767* (2019), 73–82.

[80] Gilmore, P. C., and Hoffman, A. J. A characterization of comparability graphs and of interval graphs. *Canadian Journal of Mathematics 16* (1964), 539–548.

[81] Goddard, W., Hedetniemi, S. M., Hedetniemi, S. T., and Laskar, R. Generalized subgraph-restricted matchings in graphs. *Discrete Mathematics 293*, 1 (2005), 129–138.

[82] Golumbic, M. C., Hirst, T., and Lewenstein, M. Uniquely restricted matchings. *Algorithmica 31*, 2 (Oct 2001), 139–154.

[83] Golumbic, M. C., and Lewenstein, M. New results on induced matchings. *Discrete Applied Mathematics 101*, 1 (2000), 157–165.

[84] Gomes, G., Masquio, B., Pinto, P., Santos, V., and Szwarcfiter, J. Emparelhamento desconexo é np-completo. In *Anais do VI Encontro de Teoria da Computação* (Porto Alegre, RS, Brasil, 2021), SBC, pp. 26–29.

[85] Gomes, G., Masquio, B., Pinto, P., Santos, V., and Szwarcfiter, J. Emparelhamento conexo ponderado é np-completo. In *Anais do VII Encontro de Teoria da Computação* (Porto Alegre, RS, Brasil, 2022), SBC, pp. 33–36.

[86] Gomes, G., Masquio, B., Pinto, P., Santos, V., and Szwarcfiter, J. Weighted connected matchings. In *Annals of the LAWCG 22 Workshop on Cliques in Graphs* (Curitiba, PR, Brazil, 2022).

[87] Gomes, G., Masquio, B., Pinto, P., Santos, V., and Szwarcfiter, J. Weighted connected matchings. In *Abstract book of XXI Latin Ibero-American Conference on Operations Research* (Buenos Aires, Argentina, 2022).

[88] Gomes, G. C. M., Masquio, B. P., Pinto, P. E. D., dos Santos, V. F., and Szwarcfiter, J. L. Disconnected matchings. In *Computing and Combinatorics* (Cham, 2021), C.-Y. Chen, W.-K. Hon, L.-J. Hung, and C.-W. Lee, Eds., Springer International Publishing, pp. 579–590.

[89] Gomes, G. C. M., Masquio, B. P., Pinto, P. E. D., Santos, V. F. d., and Szwarcfiter, J. L. Weighted connected matchings. In *LATIN 2022: Theoretical Informatics* (Cham, 2022), A. Castañeda and F. Rodríguez-Henríquez, Eds., Springer International Publishing, pp. 54–70.

[90] Hadwiger, H. Über eine klassifikation der streckenkomplexe. *Vierteljschr. Naturforsch. Ges. Zürich 88*, 2 (1943), 133–142.

[91] Hajebi, S., and Javadi, R. On the parameterized complexity of the acyclic matching problem. *arXiv preprint abs/2109.06004* (2021).

[92] Halappanavar, M. *Algorithms for Vertex-Weighted Matching in Graphs*. PhD thesis, Old Dominion University, 2009.

[93] Halász, G., and Sós, V. T. *Problems*. Springer Berlin Heidelberg, Berlin, Heidelberg, 1989, pp. 161–165.

[94] Hall, P. On representatives of subsets. *Journal of the London Mathematical Society s1-10*, 1 (1935), 26–30.

[95] Holyer, I. The NP-completeness of edge-coloring. *SIAM Journal on Computing 10*, 4 (Nov. 1981), 718–720.

[96] Hopcroft, J., and Karp, R. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal on Computing 2*, 4 (1973), 225–231.

[97] Horton, J. D., and Kilakos, K. Minimum edge dominating sets. *SIAM Journal on Discrete Mathematics 6*, 3 (1993), 375–387.

[98] Huang, C.-C., Kavitha, T., Mehlhorn, K., and Michail, D. Fair matchings and related problems. *Algorithmica 74*, 3 (Mar 2016), 1184–1203.

[99] Hurley, E., de Joannis de Verclos, R., and Kang, R. J. An improved procedure for colouring graphs of bounded local density. In *Proceedings of the Thirty-Second Annual ACM-SIAM Symposium on Discrete Algorithms* (USA, 2021), SODA '21, Society for Industrial and Applied Mathematics, p. 135–148.

[100] Hwang, S.-F., and Chang, G. The edge domination problem. *Discussiones Mathematicae Graph Theory 15*, 1 (1995), 51–57.

[101] Hylland, A., and Zeckhauser, R. The efficient allocation of individuals to positions. *Journal of Political economy 87*, 2 (1979), 293–314.

[102] Irving, R. W. Greedy matchings. *University of Glasgow, Computing Science Department Research Report, TR-2003-136* (2003).

[103] Irving, R. W., Kavitha, T., Mehlhorn, K., Michail, D., and Paluch, K. E. Rank-maximal matchings. *ACM Trans. Algorithms 2*, 4 (oct 2006), 602–610.

[104] Iwama, K., and Miyazaki, S. A survey of the stable marriage problem and its variants. In *International Conference on Informatics Education and Research for Knowledge-Circulating Society (icks 2008)* (2008), pp. 131–136.

[105] Jamieson, A. *Linear-Time Algorithms for Edge-Based Problems.* PhD thesis, Clemson University, Clemson, SC, USA, 2007.

[106] Jebara, T., and Shchogolev, V. B-matching for spectral clustering. In *Machine Learning: ECML 2006* (Berlin, Heidelberg, 2006), J. Fürnkranz, T. Scheffer, and M. Spiliopoulou, Eds., Springer Berlin Heidelberg, pp. 679–686.

[107] Jebara, T., Wang, J., and Chang, S.-F. Graph construction and b-matching for semi-supervised learning. In *Proceedings of the 26th Annual International Conference on Machine Learning* (New York, NY, USA, 2009), ICML '09, Association for Computing Machinery, p. 441–448.

[108] Jobson, A. S., Kézdy, A. E., and White, S. C. Connected matchings in chordal bipartite graphs. *Discrete Optimization 14* (2014), 34–45.

[109] Kalyanasundaram, B., and Pruhs, K. R. An optimal deterministic algorithm for online *b*-matching. *Theoretical Computer Science 233*, 1 (2000), 319–325.

[110] Kameda, T., and Munro, I. A O(|V|·|E|) algorithm for maximum matching of graphs. *Computing 12*, 1 (Mar 1974), 91–98.

[111] Karp, R. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, R. Miller and J. Thatcher, Eds. Plenum Press, 1972, pp. 85–103.

[112] Kasteleyn, P. W. Dimer statistics and phase transitions. *Journal of Mathematical Physics 4*, 2 (Feb. 1963), 287–293.

[113] Kavitha, T., and Shah, C. D. Efficient algorithms for weighted rank-maximal matchings and related problems. In *Algorithms and Computation* (Berlin, Heidelberg, 2006), T. Asano, Ed., Springer Berlin Heidelberg, pp. 153–162.

[114] Kikuno, T., Yoshida, N., and Kakuda, Y. The np-completeness of the dominating set problem in cubic planer graphs. *IEICE TRANSACTIONS (1976-1990) 63*, 6 (1980), 443–444.

[115] Klemz, B., and Rote, G. Linear-time algorithms for maximum-weight induced matchings and minimum chain covers in convex bipartite graphs. *Algorithmica* (Jan 2022).

[116] Ko, C., and Shepherd, F. Adding an identity to a totally unimodular matrix. *Operational Research working papers (LSEOR 94.14)* (1994).

[117] Kobler, D., and Rotics, U. Finding maximum induced matchings in subclasses of claw-free and p 5-free graphs, and in graphs with matching and induced matching of equal maximum size. *Algorithmica 37*, 4 (2003), 327–346.

[118] Kocaoglu, M., Shanmugam, K., and Bareinboim, E. Experimental design for learning causal graphs with latent variables. *Advances in Neural Information Processing Systems 30* (2017).

[119] König, D. Über graphen und ihre anwendung auf determinantentheorie und mengenlehre. *Mathematische Annalen 77*, 4 (Dec. 1916), 453–465.

[120] König, D. Gráfok és mátrixok. *Matematikai és Fizikai Lapok 38* (1931), 116–119.

[121] Kuhn, H. W. The hungarian method for the assignment problem. *Naval research logistics quarterly 2*, 1-2 (1955), 83–97.

[122] Lawler, E. *Combinatorial Optimization: Networks and Matroids*. Holt, Rinehart and Winston, 1976.

[123] Lewis, J. G., Peyton, B. W., and Pothen, A. A fast algorithm for reordering sparse matrices for parallel factorization. *SIAM J. Sci. Stat. Comput. 10*, 6 (Nov. 1989), 1146–1173.

[124] Liang, Y., and Rhee, C. Finding a maximum matching in a circular-arc graph. *Information Processing Letters 45*, 4 (1993), 185–190.

[125] Lichtenstein, D. Planar formulae and their uses. *SIAM Journal on Computing 11*, 2 (1982), 329–343.

[126] Lima, C. V., Rautenbach, D., Souza, U. S., and Szwarcfiter, J. L. Decycling with a matching. *Information Processing Letters 124* (2017), 26–29.

[127] Lima, C. V. G. C., Rautenbach, D., Souza, U. S., and Szwarcfiter, J. L. On the computational complexity of the bipartizing matching problem. *Annals of Operations Research* (Feb 2021).

[128] Lovász, L. Three short proofs in graph theory. *Journal of Combinatorial Theory, Series B 19*, 3 (1975), 269–271.

[129] Lozin, V. V. On maximum induced matchings in bipartite graphs. *Information Processing Letters 81*, 1 (2002), 7–11.

[130] Mahdian, M. On the computational complexity of strong edge coloring. *Discrete Applied Mathematics 118*, 3 (2002), 239–248.

[131] Marsh, A. *Matching Algorithms*. PhD thesis, Johns Hopkins University, 1979.

[132] Masquio, B., Pinto, P., and Szwarcfiter, J. Emparelhamentos conexos. In *Anais do V Encontro de Teoria da Computação* (Porto Alegre, RS, Brasil, 2020), SBC, pp. 1–4.

[133] Masquio, B. P. Emparelhamentos desconexos. Master's thesis, Universidade do Estado do Rio de Janeiro, 2019.

[134] Masquio, B. P., Pinto, P. E. D., and Szwarcfiter, J. L. Algoritmos lineares para emparelhamentos conexos. In *Anais Do LII Simpósio Brasileiro de Pesquisa Operacional* (João Pessoa, PB, Brazil, 2020).

[135] Masquio, B. P., Pinto, P. E. D., and Szwarcfiter, J. L. Connected matchings. Poster presented at 9th Latin American Workshop on Cliques in Graphs, 2020.

[136] Masquio, B. P., Pinto, P. E. D., and Szwarcfiter, J. L. Algoritmos lineares para emparelhamentos conexos máximos ponderados e não ponderados. In *Proceeding Series of the Brazilian Society of Computational and Applied Mathematics* (Dec. 2021), SBMAC.

[137] Mertzios, G. B., Nichterlein, A., and Niedermeier, R. A linear-time algorithm for maximum-cardinality matching on cocomparability graphs. *SIAM Journal on Discrete Mathematics 32*, 4 (Jan. 2018), 2820–2835.

[138] Micali, S., and Vazirani, V. V. An $O(\sqrt{|V|}|E|)$ algorithm for finding maximum matching in general graphs. In *21st Ann. Symp. on Foundations of Comp. Sc.* (Oct 1980), pp. 17–27.

[139] Micali, S., and Vazirani, V. V. An $\sqrt{|V|}\ |E|$ algorithm for finding maximum matching in general graphs. In *21st Annual Symposium on Foundations of Computer Science (sfcs 1980)* (Oct 1980), pp. 17–27.

[140] Michael D. Plummer, Michael Stiebitz, B. T. On a special case of hadwiger's conjecture. *Discussiones Mathematicae Graph Theory 23*, 2 (2003), 333–363.

[141] Michail, D. Reducing rank-maximal to maximum weight matching. *Theoretical Computer Science 389*, 1 (2007), 125–132.

[142] Mishra, S. On the maximum uniquely restricted matching for bipartite graphs. *Electronic Notes in Discrete Mathematics 37* (2011), 345–350. LAGOS'11 – VI Latin-American Algorithms, Graphs and Optimization Symposium.

[143] Moser, H., and Sikdar, S. The parameterized complexity of the induced matching problem. *Discrete Applied Mathematics 157*, 4 (2009), 715–727.

[144] Okamoto, Y., Uehara, R., and Uno, T. Counting the number of matchings in chordal and chordal bipartite graph classes. In *Graph-Theoretic Concepts in Computer Science* (Berlin, Heidelberg, 2010), C. Paul and M. Habib, Eds., Springer Berlin Heidelberg, pp. 296–307.

[145] Orlovich, Y., Finke, G., Gordon, V., and Zverovich, I. Approximability results for the maximum and minimum maximal induced matching problems. *Discrete Optimization 5*, 3 (2008), 584–593.

[146] Orlovich, Y. L., and Zverovich, I. E. Maximal induced matchings of minimum/maximum size. *Technical report, DIMACS TR 2004-26* (2004).

[147] Paluch, K. Capacitated rank-maximal matchings. In *Algorithms and Complexity* (Berlin, Heidelberg, 2013), P. G. Spirakis and M. Serna, Eds., Springer Berlin Heidelberg, pp. 324–335.

[148] Panda, B. S., and Chaudhary, J. Acyclic matching in some subclasses of graphs. In *Combinatorial Algorithms* (Cham, 2020), L. Gasieniec, R. Klasing, and T. Radzik, Eds., Springer International Publishing, pp. 409–421.

[149] Panda, B. S., and Pandey, A. On the complexity of minimum cardinality maximal uniquely restricted matching in graphs. In *Theoretical Computer Science and Discrete Mathematics* (Cham, 2017), S. Arumugam, J. Bagga, L. W. Beineke, and B. Panda, Eds., Springer International Publishing, pp. 218–227.

[150] Panda, B. S., Pandey, A., Chaudhary, J., Dane, P., and Kashyap, M. Maximum weight induced matching in some subclasses of bipartite graphs. *Journal of Combinatorial Optimization 40*, 3 (Oct 2020), 713–732.

[151] Pandey, A., Panda, B. S., Dane, P., and Kashyap, M. Induced matching in some subclasses of bipartite graphs. In *Algorithms and Discrete Applied Mathematics* (Cham, 2017), D. Gaur and N. Narayanaswamy, Eds., Springer International Publishing, pp. 308–319.

[152] Petersen, J. Die theorie der regulären graphs. *Acta Mathematica 15*, 1 (1891), 193–220.

[153] Pinar, A., Chow, E., and Pothen, A. Combinatorial algorithms for computing column space bases that have sparse inverses.

[154] Plummer, M., and Lovász, L. *Matching theory*. ISSN. Elsevier Science, 1986.

[155] Pothen, A., and Fan, C.-J. Computing the block triangular form of a sparse matrix. *ACM Transactions on Mathematical Software (TOMS) 16*, 4 (1990), 303–324.

[156] Protti, F., and Souza, U. S. Decycling a graph by the removal of a matching: new algorithmic and structural aspects in some classes of graphs. *Discrete Mathematics & Theoretical Computer Science vol. 20 no. 2* (Nov. 2018).

[157] Pulleyblank, William R. *Faces of Matching Polyhedra*. PhD thesis, University of Waterloo, 1973.

[158] Richey, M., and Parker, R. G. Minimum-maximal matching in series-parallel graphs. *European journal of operational research 33*, 1 (1988), 98–105.

[159] Robertson, N., and Seymour, P. D. Graph minors. II. algorithmic aspects of tree-width. *Journal of Algorithms 7*, 3 (1986), 309 – 322.

[160] Salavatipour, M. R. A polynomial time algorithm for strong edge coloring of partial k-trees. *Discrete Applied Mathematics 143*, 1 (2004), 285–291.

[161] Schaefer, T. J. The complexity of satisfiability problems. In *Proceedings of the Tenth Annual ACM Symposium on Theory of Computing* (New York, NY, USA, 1978), STOC '78, Association for Computing Machinery, p. 216–226.

[162] Sharma, G., Mazumdar, R. R., and Shroff, N. B. On the complexity of scheduling in wireless networks. In *Proceedings of the 12th annual international conference on Mobile computing and networking* (2006), pp. 227–238.

[163] Shen, H., and Liang, W. Efficient enumeration of all minimal separators in a graph. *Theoretical Computer Science 180*, 1-2 (1997), 169–180.

[164] Spencer, T. H., and Mayr, E. W. Node weighted matching. In *Automata, Languages and Programming* (Berlin, Heidelberg, 1984), J. Paredaens, Ed., Springer Berlin Heidelberg, pp. 454–464.

[165] Spinrad, J. *Efficient Graph Representations*, vol. 19. American Mathematical Soc., 2003.

[166] Srinivasan, A., Madhukar, K., Nagavamsi, P., Rangan, C. P., and Chang, M.-S. Edge domination on bipartite permutation graphs and cotriangulated graphs. *Information Processing Letters 56*, 3 (1995), 165–171.

[167] Steiner, G., and Yeomans, J. A linear time algorithm for maximum matchings in convex, bipartite graphs. *Computers & Mathematics with Applications 31*, 12 (1996), 91–96.

[168] Stockmeyer, L. J., and Vazirani, V. V. Np-completeness of some generalizations of the maximum matching problem. *Information Processing Letters 15*, 1 (1982), 14–19.

[169] Surynek, P. Compact representations of cooperative path-finding as sat based on matchings in bipartite graphs. In *2014 IEEE 26th International Conference on Tools with Artificial Intelligence* (2014), IEEE, pp. 875–882.

[170] Tabatabaee, V., Georgiadis, L., and Tassiulas, L. Qos provisioning and tracking fluid policies in input queueing switches. *IEEE/ACM Transactions on Networking 9*, 5 (2001), 605–617.

[171] Tamir, A., and Mitchell, J. A maximum b-matching problem arising from median location models with application to the roommates problem. *Mathematical Programming 80* (12 1998), 171–194.

[172] Tarjan, R. E., and Yannakakis, M. Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. *SIAM J. Comput. 13*, 3 (July 1984), 566–579.

[173] Tutte, W. T. The factorization of linear graphs. *Journal of the London Mathematical Society s1-22*, 2 (1947), 107–111.

[174] Uno, T. Algorithms for enumerating all perfect, maximum and maximal matchings in bipartite graphs. In *Algorithms and Computation* (Berlin, Heidelberg, 1997), H. W. Leong, H. Imai, and S. Jain, Eds., Springer Berlin Heidelberg, pp. 92–101.

[175] Uno, T. A fast algorithm for enumerating bipartite perfect matchings. In *Algorithms and Computation* (Berlin, Heidelberg, 2001), P. Eades and T. Takaoka, Eds., Springer Berlin Heidelberg, pp. 367–379.

[176] Vadhan, S. P. The complexity of counting in sparse, regular, and planar graphs. *SIAM Journal on Computing 31*, 2 (2001), 398–427.

[177] Valiant, L. The complexity of computing the permanent. *Theoretical Computer Science 8*, 2 (1979), 189–201.

[178] Valiant, L. G. The complexity of enumeration and reliability problems. *SIAM Journal on Computing 8*, 3 (1979), 410–421.

[179] Vizing, V. G. On an estimate of the chromatic class of a p-graph. *Discret Analiz 3* (1964), 25–30.

[180] Witzgall, C., and Zahn, C. Modification of edmonds' maximum matching algorithm. *Journal of Research of the National Bureau of Standards Section B Mathematics and Mathematical Physics 69B*, 1 and 2 (Jan. 1965), 91.

[181] Wu, J., and Lin, W. The strong chromatic index of a class of graphs. *Discrete Mathematics 308*, 24 (2008), 6254–6261.

[182] Yang, W., and Wu, B. Proof of a conjecture on the strong chromatic index of halin graphs. *Discrete Applied Mathematics 302* (2021), 92–102.

[183] Yannakakis, M., and Gavril, F. Edge dominating sets in graphs. *SIAM Journal on Applied Mathematics 38*, 3 (1980), 364–372.

[184] Yu, M.-S., and Yang, C.-H. An o(n) time algorithm for maximum matching on cographs. *Information Processing Letters 47*, 2 (1993), 89–93.

[185] Yuan, Y. Residence exchange wanted: A stable residence exchange problem. *European Journal of Operational Research 90*, 3 (1996), 536–546.

[186] Yuster, R. Maximum matching in regular and almost regular graphs. *Algorithmica 66*, 1 (Feb. 2012), 87–92.

[187] Zhao, X., and Zhou, X. Strong chromatic index of graphs: a short survey. *International Journal of Engineering Research & Science 1* (2015), 1–8.

[188] Zito, M. Induced matchings in regular graphs and trees. In *International Workshop on Graph-Theoretic Concepts in Computer Science* (1999), Springer, pp. 89–101.

[189] Zito, M. *Randomised techniques in combinatorial algorithmics.* PhD thesis, Department of Computer Science, 1999.