Matching Problems with Additional Resource Constraints

Dissertation

zur Erlangung des akademischen Grades
eines Doktors der Naturwissenschaften (Dr. rer. nat.)

Dem Fachbereich IV der Universität Trier
vorgelegt von

Dirk Johannes Thomas

Trier, 2015

Eingereicht am 29. Oktober 2015

Gutachter: Prof. Dr. Sven de Vries
Prof. Dr. Sven O. Krumke

Für meine Eltern
Sanni und Werner.

Contents

Zusammenfassung (German Summary)

Matchingprobleme mit zusätzlichen Ressourcenbeschränkungen stellen eine Verallgemeinerung klassischer Matchingprobleme dar. Damit eine Menge von Kanten ein zulässiges Matching für ein solches Problem ist, muss sie neben den Matchingbedinungen noch die Anforderungen der zusätzlichen Ressourcenbeschränkungen erfüllen. Im Fokus dieser Arbeit stehen Matchingprobleme mit zwei Arten zusätzlicher Ressourcenbeschränkungen: Das *couple constrained matching problem* und das *level constrained matching problem*. Im couple constrained matching problem fordern die zusätzlichen Ressourcenbeschränkungen, dass für eine Menge von Kantenpaaren (sogenannten *couples*) die Kanten eines Paares entweder beide im Matching sind oder beide nicht im Matching sind. Im level constrained matching problem kommt exakt eine zusätzliche Ressourcenbeschränkung vor. Sie gibt die Anzahl von sogenannten *on-level* Kanten in einem Matching vor. In einem bipartiten Graphen, dessen Knoten mit Index notiert sind, sind dies jene Kanten, deren zwei Endknoten den gleichen Index haben.

Zu Beginn werden grundlegende Formulierungen und Eigenschaften von Matchingproblemen wiederholt, die für das Verständnis dieser Arbeit von Bedeutung sind. Darauffolgend werden die Formulierungen der beiden Hauptprobleme vorgestellt. Es wird für beide Probleme gezeigt, dass deren Matchingvarianten maximales Matching, vollständiges Matching und perfektes Matching polynomiell äquivalent sind. Als zusätzliche Variante wird das Assignmentproblem mit beiden Arten von Ressourcenbeschränkungen formuliert. Des Weiteren werden Probleme aus der Literatur vorgestellt, die in Bezug zu den hier behandelten Matchingproblemen und deren Ressourcenbeschränkungen stehen.

Ein zentrales Ergebnis bezüglich des couple constrained matching problem ist, dass dieses Problem NP-schwer ist. Dies wird durch die Verallgemeinerung einer Komplexitätsaussage zum Problem weighted matching with bonds aus der Literatur bewiesen. Darüber hinaus geht aus den durchgeführten Reduktionsschritten des Beweises hervor, dass das couple constrained matching problem auch dann NP-schwer ist, wenn es auf bipartite Kreisgraphen beschränkt wird.

Ebenso befasst sich diese Arbeit mit Komplexitätsaspekten zum level constrained matching problem. Dazu wird das Problem als perfektes Matchingproblem untersucht. Es werden drei kombinatorische Optimierungsprobleme aus der Literatur vorgestellt, deren Komplexität unbekannt ist, und die alle polynomiell äquivalent zum level constrained perfect matching problem sind. Eines dieser drei Probleme ist das resource constrained perfect matching problem. Dieses Problem ist ein perfektes Matchingproblem mit variabler Anzahl zusätzlicher Ressourcenbeschränkungen. Diese geben für beliebige Teilmengen der Kanten obere Schranken für die Anzahl von Matchingkanten aus diesen Teilmengen vor. Anhand dieses Problems wird dargestellt, welchen Einfluss fixe und variable Parameter auf die Problemkomplexität haben können. Dazu wird für verschiedene Kombinationen von fixen und variablen Eingabeparametern des Problems gezeigt, dass sie die polynomielle Lösbarkeit oder die NP-Schwere des Problems implizieren. Dies erlaubt es auch, das level constrained perfect matching problem hinsichtlich seiner Komplexität innerhalb dieser Problemvarianten einzuordnen.

Ein Kapitel dieser Arbeit befasst sich damit, die beiden hier untersuchten Arten von Ressourcenbeschränkungen zueinander in Bezug zu setzen. Zunächst wird gezeigt, dass das level constrained perfect matching problem ein Spezialfall des couple constrained perfect matching problem ist. Als wesentlicher Bestandteil dieser Thematik wird dann die Komplexität des sogenannten *couple and level constrained matching problem with on-level couples* bewiesen. Dieses Problem enthält beide Arten von Ressourcenbeschränkungen, d.h. es enthält sowohl couple constraints als auch eine level constraint. Mittels polynomieller Reduktion des Cliquenproblems wird gezeigt, dass dieses Problem NP-vollständig ist. Der zugehörige Beweis basiert auf einer modularen Konstruktion und Analyse der Instanz des couple and level constrained matching problem with on-level couples in der Reduktion. Diesem Komplexitätsergebnis kommt eine besondere Bedeutung zu, da das Problem ohne die level constraint leicht zu lösen ist. Somit wird auch gezeigt, welchen Einfluss die level constraint auf eine leicht zu lösende – und insbesonders polynomiell lösbare – Variante eines ressourcenbeschränkten Matchingproblems haben kann.

Ein weiterer Teil dieser Arbeit widmet sich den Polytopen von Matchingproblemen mit zusätzlichen Ressourcenbeschränkungen. Für das Polytop des relaxierten level constrained perfect matching problem wird eine Charakterisierung seiner nicht ganzzahligen Ecken erarbeitet. Dazu wird eine Menge linearer Ungleichungen eingeführt für die gezeigt wird, dass sie alle nicht ganzzahligen Ecken dieses Polytops von der convexen Hülle aller ganzzahligen Lösungen separieren. Zudem wird dargelegt, dass für jede nicht ganzzahlige Ecke eine separierende Ungleichung in polynomiell vielen Schritten gefunden werden kann.

Zur Lösung des level constrained matching problem werden zwei neue Algorithmen entwickelt. Der erste Algorithmus ist ein Approximationsalgorithmus für das level constrained matching problem auf level Graphen. Dieser hat die Eigenschaft, dass die von ihm bestimmten Matchings die level constraint erfüllen und mindestens $z^* - 1$ Kanten enthalten, wobei z^* die Kardinalität der Optimallösung ist. Der zweite Algorithmus ist der *Objective Branching Algorithmus*. Dieser löst das equality constrained perfect matching problem – ein perfektes Matchingproblem mit verallgemeinerter Variante der level constraint – exakt. Er nutzt dabei aus, dass das gewichtete perfekte Matchingproblem ohne zusätzliche Ressourcenbeschränkung polynomiell lösbar ist. Die Idee hinter dem Algorithmus ist, dass eine Baumstruktur von Teilproblemen aufgebaut wird. Mit Hilfe dieser Teilprobleme werden mögliche Lösungen des equality constrained perfect matching problem bestimmt. Experimentelle Ergebnisse einer Implementierung des Objective Branching Algorithmus finden sich im Anhang dieser Arbeit.

Preface

In the wide field of combinatorial optimization problems, the matching problem is one of the most investigated and best understood problems. In 1965 Jack Edmonds developed his “blossom algorithm” for determining a maximum cardinality matching in graphs in polynomial-time. From about that time onwards, the importance of matching problems as a tool for modeling and solving many real-world problems became especially apparent. From a complexity point of view, his work showed that the matching problem belongs to the important class P of polynomially solvable problems.

Motivation of this work

The possibility of imposing additional constraints to the classical matching problem proves very useful in modeling practical problems. It allows the creation of models that represent the actual problem more accurately. Moreover, the additional constraints might even be indispensable for modeling certain conditions. When matching problems with different types of additional constraints are investigated, new combinatorial and graph theoretical approaches to the problem need to be developed. Further, these problems give rise to investigations of their polytopal structure and lead to the challenge of finding novel solution algorithms for the specific matching variant.

When tackling the task of finding a solution to a matching problem with additional constraints, knowing about the complexity of that problem is of great advantage. If the problem is NP-hard, then it is preferable (under the widely assumed hypothesis that $P \neq NP$) to aim for a polynomial approximation algorithm rather than for a polynomial algorithm which gives an exact solution to the problem. From a theoretical point of view, investigations on the complexity of a matching problem with additional constraints often reveal a connection to other combinatorial problems and shows how the problems can be transformed into each other.

In general, additional side constraints of matching problems are referred to as resource constraints. A very basic type of resource constraints are equality constraints. We will distinguish whether they are imposed on matching problems as a set of additional constraints or as an individual constraint. Either way, the additional constraints themselves are part of the problem input. When imposed as a set of constraints, the total number of additional equality constraints is a significant part of the characteristic of the problem. When imposed as a single constraint, it is the number of edges occurring in the equality which is of particular interest.

In this work, we consider matching problems with two types of additional equality constraints that represent these two opposed cases. The first one is a matching problem which has imposed a set of additional equality constraints. Each constraint demands that for a given pair of edges either both edges are in the matching or none of them is in the matching. This problem type represents the case of a variable number of additional equality constraints with each individual equality having the lowest possible number of variables while not being trivial. The second one is a matching problem which has imposed a single equality constraint. This constraint demands that an exact number of edges in the matching are elements of a special subset of the edges in the graph. The content of this subset depends on the graph of the problem instance. This problem type represents the case of a minimum number of additional constraints where it is the number of variables in this constraint which is variable.

It is worth mentioning that the complexity of matching problems with additional equality constraints is dependent on the combination of the basic matching problem with these constraints. Hence, the effect of the additional constraints on the basic matching problem is of particular interest. The investigations in this work aim at a better understanding of complexity-related aspects of matching problems with additional equality constraints. For that, it is important to mention that throughout this work we follow the assumption that $\mathrm{P} \neq \mathrm{NP}$.

Structure of this work

In Chapter 1, we summarize elementary facts about classical matching problems that are of relevance to this work. We recapitulate formulations of variants of matching problems and briefly recall well-known transformation techniques and algorithms to solve them. Further, definitions and basic properties of the matching polytope and the perfect matching polytope are given.

In Chapter 2, the two main problems of this work – the couple constrained matching problem and the level constrained matching problem – are introduced. Both problems belong to the class of resource constrained matching problems. For each of the two problems the respective problem formulation is given in form of an integer linear program. Further, we show the equivalence of different optimization variants of the respective problems and present the assignment problems with corresponding side constraints. In what follows, the implications of fixed parameters in the additional side constraints we consider are discussed. This is done in the more general context of combinatorial optimization problems, rather than restricting the assertions specifically to matching problems. The chapter is concluded by presenting combinatorial optimization problems from the literature which are related to our resource constrained matching problems.

Chapter 3 deals with the complexity of the couple constrained matching problem. We start this chapter with a generalization of a known complexity result for the problem of weighted matching with bonds. After showing some intermediate results, we finally prove the new result that the couple constrained matching problem is NP-hard on bipartite cycle graphs.

In Chapter 4, we examine complexity related aspects of the level constrained matching problem. In this context we show that the level constrained perfect matching problem is polynomially equivalent to three other combinatorial optimization problems from the literature. One of these problems is the so-called restricted perfect matching problem, which is a perfect matching problem whose additional side constraints are upper bound constraints rather than equality constraints. For different combinations of fixed and variable parameters in these side constraints we investigate their effect on the complexity of the problem. We show which conditions are sufficient for the problem to be NP-hard and which are sufficient for it to be polynomially solvable. At the end of this chapter, the complexity of the assignment problem with an additional equality constraint is investigated.

In Chapter 5, couple constraints and the level constraint are associated with each other. We first show that the level constrained perfect matching problem is a special case of the couple constrained perfect matching problem. Then, we show that the level constraint can be sufficient for making a polynomially solvable problem NP-hard when being imposed on that problem. To this end, we introduce the couple and level constrained matching problem with on-level couples and prove that its decision version is NP-complete. This problem is a matching problem with a special case of couple constraints together with a level constraint imposed on it. Without the additional level constraint, the corresponding problem is polynomially solvable.

The topic of Chapter 6 is the polyhedral structure of resource constrained matching problems. We present facet defining inequalities for the polytope corresponding to a special case of the couple constrained assignment problem, as given in the literature. For the polytope corresponding to the level constrained perfect matching problem we take a closer look at its non-integral vertices. We develop a set of valid inequalities which separates all non-integral vertices of the polytope from the convex hull of its integral points. Further, we prove that for a given non-integral vertex of the polytope a corresponding inequality which separates this vertex from the convex hull of integral points can be found in polynomial time.

In Chapter 7, we deal with questions regarding the calculation of solutions of resource constrained matching problems. We develop a new polynomial approximation algorithm for the level constrained matching problem, which returns solutions whose size is at most one less than the size of an optimal solution. We then describe a new algorithm for exactly solving the perfect matching problem with an additional equality constraint. The algorithm makes use of the fact that the perfect matching problem without an additional side constraint is polynomially solvable.

Chapter 8 summarizes the main results of this work.

In the Appendix, a brief summary of the terminology and principles of computational complexity is given. Further, it lists algorithms and computational results which are mentioned in this work but are not crucial for the understanding of related topics. Finally, a list of all problem formulations appearing in this work can be found in the Appendix. It may be advisable to use this part for looking up specific formulations.

1. Matchings – Introduction and Basic Concepts

In this chapter, we recall those problem formulations and results concerning matching problems which become relevant in the course of this work. This includes the formulation of the classical matching problem and its variants, standard methods to solve these problems and basic properties of the corresponding polytopes. For most of the considerations in this chapter we distinguish between the bipartite case and the general (nonbipartite) case.

We assume the reader to be familiar with the basic notations used in combinatorial optimization. The formulations of the flow problems and the basic matching problems in this chapter are as given by Ahuja et. al in [4]. For a comprehensive overview of the wide field of matching problems in general we refer to the work of Ahuja et. al in [4] and Schrijver in [49].

The chapter starts with an introduction of two types of flow problems in Section 1.1. We formulate the maximum flow problem and the minimum cost flow problem with regards to their importance as solution methods for bipartite matching problems.

In Section 1.2, we recall formulations of different variants of matching problems, including the classical matching problem, its weighted version and the perfect and the complete matching problem. Further, the assignment problem is formulated in terms as used in this work.

Section 1.3 deals with polynomial solution methods of the matching problem and its variants from Section 1.2. These methods are based on transforming the problems into flow problems and on the usage of the Hungarian algorithm.

The perfect matching polytope and the matching polytope are considered in Section 1.4. We recall the properties of the polytopes which are of particular importance for this work.

The chapter is concluded with Section 1.5, where the 3-dimensional matching problem is introduced. The problem is formulated in terms of a hypergraph and its complexity is stated.

1.1. Flow problems

1.1.1. The maximum flow problem

Let $D = (N, A)$ be a directed graph with a source node $s \in N$ and a sink node $t \in N$. Further, let u_{ij} be the nonnegative arc capacity of arc (i, j) for all $(i, j) \in A$.

Problem Formulation 1.1 (Maximum flow). *The task of the* maximum flow problem *is to find a feasible flow of maximum value. It is formulated as the linear program*

$$\max \quad v \tag{1.1}$$

$$s.t. \quad \sum_{j:(i,j)\in A} x_{ij} - \sum_{j:(j,i)\in A} x_{ji} = \begin{cases} v & \text{if } i = s, \\ 0 & \text{if } i \in N \setminus \{s,t\}, \\ -v & \text{if } i = t \end{cases} \tag{1.2}$$

$$0 \le x_{ij} \le u_{ij} \qquad \forall\ (i,j) \in A. \tag{1.3}$$

The constraints (1.2) are called *flow preservation constraints.* Each vector $x \in \mathbb{R}^A$ which satisfies (1.2) and (1.3) is called a *feasible flow* in D. The value of the variable v is the corresponding *flow value.* Each component x_{ij} is the amount of flow carried by arc (i,j).

For a list of algorithms and corresponding running-times for the maximum flow problem see Appendix B. Among these algorithms there are algorithms having a polynomial running-time, which implies that the maximum flow problem is polynomially solvable. The extension of this result to the integer case is shown in [4].

Theorem 1.2. *The maximum flow problem is polynomially solvable. If all arc capacities are integer, the problem has an integer solution which can also be found in polynomial-time.*

1.1.2. The minimum cost flow problem

A generalization of the maximum flow problem is the minimum cost flow problem. Additionally to the capacity values of arcs in A there is a cost value $c_{ij} \in \mathbb{R}$ for all arcs $(i,j) \in A$. Furthermore, a feasible flow does not necessarily need to be balanced at each node in $N \setminus \{s,t\}$. Instead, the amount of flow emanating from a node minus the amount of flow entering the node must equal a given number. Let $b(i)$ be this number for all nodes $i \in N$. If $b(i) > 0$ it is called the *supply* of node i, if $b(i) < 0$ it is called the *demand* of node i.

Problem Formulation 1.3 (Minimum cost flow). *The task of the* minimum cost flow problem *is to find a feasible b-flow of minimum cost. It is formulated as the linear program*

$$\min \quad \sum_{(i,j)\in A} c_{ij} x_{ij} \tag{1.4}$$

$$s.t. \quad \sum_{j:(i,j)\in A} x_{ij} - \sum_{j:(j,i)\in A} x_{ji} = b(i) \qquad \forall\ i \in N \tag{1.5}$$

$$0 \le x_{ij} \le u_{ij} \qquad \forall\ (i,j) \in A. \tag{1.6}$$

Each vector $x \in \mathbb{R}^A$ which satisfies (1.5) is called a *b-flow* in D. If it additionally fulfills (1.6) it is a *feasible b-flow.* For a list of algorithms and corresponding running-times for the minimum cost flow problem see Appendix B. Among these algorithms there are algorithms having a polynomial running-time, which implies that the minimum cost flow problem is polynomially solvable. The extension of this result to the integer case is shown in [4].

Theorem 1.4. *The minimum cost flow problem is polynomially solvable. If all capacities and supply/demand values are integer, the problem has in integer solution which can also be found in polynomial-time.*

1.2. Formulations of matching problems

Let $G = (V, E)$ be an undirected graph. A *matching* in G is a set of pairwise node-disjoint edges in E. We identify a matching M in G with its incidence vector $\chi^M \in \{0,1\}^E$, where

$$\chi_e^M = \begin{cases} 1 & \text{if } e \in M, \\ 0 & \text{otherwise} \end{cases}$$

for all $e \in E$. In the case that the edges in E are denoted with indices, we simplify the notation by writing χ_i^M instead of $\chi_{e_i}^M$ for all $e_i \in E$. Given an edge $[u_i, v_j] \in E$, we also use the notations $\chi_{[u_i,v_j]}^M$ and χ_{ij}^M equivalently.

Given a matching M in G, a node $v \in V$ is called *M-covered* (or *covered by M*) if there exists an edge $[v, w] \in M$. Otherwise, the node v is called *M-exposed.*

1.2.1. Matching problems on general graphs

Problem Formulation 1.5 (Matching)**.** *The* matching problem *is the problem of finding a matching of maximum cardinality. It is formulated as the integer linear program*

$$\begin{aligned} \max \quad & \sum_{e \in E} x_e && && (1.7) \\ s.t. \quad & \sum_{e \in \delta(v)} x_e \leq 1 && \forall\, v \in V && (1.8) \\ & x_e \in \{0,1\} && \forall\, e \in E, && (1.9) \end{aligned}$$

where $\delta(v)$ denotes the set of edges incident to node v for all $v \in V$.

An optimal solution $x \in \{0,1\}^E$ of a matching problem on G then is the incidence vector of a matching of maximum cardinality in G. For a given matching M in G, we say that two edges in E have the same *matching activity* if either both edges are in M or both are not.

A matching M in the graph G is called a *perfect matching* if it covers all nodes in G.

Problem Formulation 1.6 (Perfect matching)**.** *The* perfect matching problem *is the problem of finding a matching M in G such that all nodes in G are covered by an edge* *in M. It is formulated as the problem of finding a feasible solution to the system*

$$\begin{aligned}\sum_{e\in\delta(v)} x_e &= 1 &&\forall\, v \in V\\ x_e &\in \{0,1\} &&\forall\, e \in E.\end{aligned}$$

The (perfect) matching problem (on bipartite and nonbipartite graphs) and the assignment problem (defined in Section 1.2.3) are the main problem types on which the resource constrained matching problems we investigate in this work are based.

When each edge $e \in E$ has an assigned weight $c_e \in \mathbb{R}$, then the matching problem can be generalized by searching for a matching of maximum weight rather than one of maximum cardinality. This yields the following problem:

Problem Formulation 1.7 (Weighted matching)**.** *The* weighted matching problem *is the problem of finding a matching M in G which is of maximum weight. It is formulated as the integer linear program*

$$\begin{aligned}\max \quad & \sum_{e\in E} c_e x_e && (1.10)\\ \textit{s.t.} \quad & \textit{(1.8), (1.9).}\end{aligned}$$

1.2.2. Matching problems on bipartite graphs

When the matching problem is stated on a bipartite graph $G = (U \cup V, E)$, constraints (1.8) are replaced by the constraints

$$\begin{aligned}\sum_{e\in\delta(u)} x_e &\le 1 &&\forall\, u \in U, && (1.11)\\ \sum_{e\in\delta(v)} x_e &\le 1 &&\forall\, v \in V. && (1.12)\end{aligned}$$

Problem Formulation 1.8 (Bipartite matching)**.** *The* bipartite matching problem *is the problem of finding a matching of maximum cardinality in a bipartite graph. It is formulated as the integer linear program*

$$\begin{aligned}\max \quad & \sum_{e\in E} x_e && && (1.13)\\ \textit{s.t.} \quad & \sum_{e\in\delta(u)} x_e \le 1 && \forall\, u \in U && (1.14)\\ & \sum_{e\in\delta(v)} x_e \le 1 && \forall\, v \in V && (1.15)\\ & x_e \in \{0,1\} && \forall\, e \in E. && (1.16)\end{aligned}$$

In the case of a *perfect matching problem* on a bipartite graph, constraints (1.14) and (1.15) are considered as equalities.

Problem Formulation 1.9 (Bipartite perfect matching)**.** *The* bipartite perfect matching problem *is the problem of finding a perfect matching in a bipartite graph. It is formulated as the problem of finding a feasible solution to the system*

$$\begin{aligned} \sum_{e\in\delta(u)} x_e &= 1 && \forall\ u \in U \\ \sum_{e\in\delta(v)} x_e &= 1 && \forall\ v \in V \\ x_e &\in \{0,1\} && \forall\ e \in E. \end{aligned}$$

A perfect matching in a bipartite graph $G = (U \cup V, E)$ can only exist if the sizes of the two color classes in G coincide, i.e. $|U| = |V|$. We next consider *complete* matchings, which are matchings that cover all nodes in the smaller of the two sets U and V.

Problem Formulation 1.10 (Complete matching)**.** *The* complete matching problem *is the problem of finding a matching M in the bipartite graph $G = (U \cup V, E)$ such that all nodes in the smaller of the two color classes U and V are covered by M. Assuming that $|U| \leq |V|$, it is formulated as the problem of finding a feasible solution to the system*

$$\begin{aligned} \sum_{e\in\delta(u)} x_e &= 1 && \forall\ u \in U \\ \sum_{e\in\delta(v)} x_e &\leq 1 && \forall\ v \in V \\ x_e &\in \{0,1\} && \forall\ e \in E. \end{aligned}$$

In contrast to the matching problem and the perfect matching problem, the complete matching problem is defined only on bipartite graphs.

For all kinds of bipartite matching problems presented in this section there exist the corresponding weighted problem variants as well. The objective of these problems is as (1.10), where $c_e \in \mathbb{R}$ is the weight of the edge e for all $e \in E$. As a special case of a weighted bipartite matching problem we consider the assignment problem.

1.2.3. The assignment problem

Following the formulations of Burkard et. al [11] and Schrijver [49], the assignment problem is a minimum cost perfect matching problem on a complete bipartite graph $K_{n,n} = (U \cup V, E)$. Hence, it generalizes the bipartite perfect matching problem.

Let c_{ij} be the nonnegative cost of the edge $[i,j]$ for all $[i,j] \in E$. In the context of an assignment problem, an edge $[i,j]$ in the graph $K_{n,n}$ with $i \in U$ and $j \in V$ will also be represented by the formulation that row i can be assigned to column j. This comes from the interpretation of the graph $K_{n,n}$ as a lattice of size $n \times n$.

Problem Formulation 1.11 (Assignment)**.** *The* assignment problem *(AP) is stated as the following integer linear program:*

$$\begin{aligned} \min \quad & \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} && && (1.17)\\ s.t. \quad & \sum_{j=1}^{n} x_{ij} = 1 && \forall\ i = 1,\dots,n && (1.18)\\ & \sum_{i=1}^{n} x_{ij} = 1 && \forall\ j = 1,\dots,n && (1.19)\\ & x_{ij} \in \{0,1\} && \forall\ i,j = 1,\dots,n. && (1.20) \end{aligned}$$

Problem (1.17) – (1.20) is a *balanced* problem, i.e. it requires a graph with two color classes of same size. In the *unbalanced* case, i.e. the case where $|U| \neq |V|$, the problem corresponding to the assignment problem is to find a complete matching which is of minimum cost. We are considering only balanced assignment problems, as problems on a graph $K_{m,n}$ with $m < n$ can be easily transformed into assignment problems on the graph $K_{n,n}$.

Further, the restriction of the costs c_{ij} to be nonnegative can be done without loss of generality. To see this, we consider a cost vector c' whose components may also be negative. Let x' be an optimal solution to (1.17) – (1.20) with cost vector c'. Let $c'_{\min} := \min\{c'_{ij} \mid i,j = 1,\dots,n\}$. We define a cost vector c as $c_{ij} := c'_{ij} - c'_{\min}$. All components of c are nonnegative. Now, we consider the assignment problem (1.17) – (1.20) with cost vector c. As each feasible solution x contains exactly n components which are of value 1 and all other components are of value 0, it holds that

$$\begin{aligned} &\min \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij}\\ =&\min \sum_{i=1}^{n}\sum_{j=1}^{n} (c'_{ij} - c'_{\min})x_{ij}\\ =&\min \sum_{i=1}^{n}\sum_{j=1}^{n} c'_{ij}x_{ij} - nc'_{\min}. \end{aligned}$$

As $nc'_{\min}$ is a constant, the assignment problem with cost vector c has the same set of optimal solutions as the assignment problem with cost vector c'. This way all problem instances with negative costs can be transformed into equivalent instances with nonnegative costs.

1.3. Solution methods for matching problems

1.3.1. Solving bipartite matching problems

A standard technique for solving matching problems on bipartite graphs is the transformation of these problems into maximum flow problems (see [4], [34]), which is described next. The corresponding instance $(D = (N, A), s, t, u)$ of the maximum flow problem (see Section 1.1.1) is constructed as follows:

$$N = U \cup V \cup \{s, t\}, \tag{1.21}$$
$$A = \{(s, i) \mid i \in U\} \cup \{(j, t) \mid j \in V\} \cup \{(i, j) \mid [i, j] \in E \text{ with } i \in U, j \in V\}, \tag{1.22}$$
$$u_{ij} = 1 \quad \text{for all } (i, j) \in A. \tag{1.23}$$

There is an equivalence of integral flows in D and matchings in G, which can be seen when defining an edge $[i, j]$ with $i \in U$ and $j \in V$ to be in a matching if and only if the corresponding arc $(i, j) \in A$ carries 1 unit of flow. Thus, each feasible integral flow of value v in D corresponds to a matching of size v in the graph G, and vice versa. This implies that a maximum integral flow in D corresponds to a maximum matching in G. As the integer maximum flow problem is polynomially solvable (see Theorem 1.2), the bipartite matching problem is polynomially solvable as well:

Theorem 1.12. *Let $G = (U \uplus V, E)$ be a bipartite graph. The matching problem on G is polynomially solvable.*

1.3.2. Solving weighted bipartite matching problems

Let $G = (U \uplus V, E)$ be a bipartite graph and let $c \in \mathbb{R}^E$ be a vector of edge weights. As the weighted matching problem aims at determining a matching which is of maximum weight, we can assume that all weights c_e are nonnegative.

Transformation into minimum cost flow problem

We first present the approach of transforming the problem into a minimum cost flow problem. The digraph $D = (N, A)$ with source and sink nodes s and t and arc capacities u_{ij} are defined as in (1.21) – (1.23) in Section 1.3.1. In addition, each arc $(i, j) \in A$ with $i \in U$ and $j \in V$ is assigned the cost $-c_{ij}$. All other arcs in A, i.e. the arcs emanating from s or ending in t, are assigned a cost value of 0. We define $b(k) = 0$ for all $k \in U \cup V$ and $b(s) = r, b(t) = -r$, where r is a non-fixed variable.

Due to the equivalence of integral flows in D and matchings in G, each feasible integral flow of total cost $-z$ in D corresponds to a matching of size r and weight z in the graph G. We solve the minimum cost flow problem for each $r = 1, \ldots, \min\{|U|, |V|\}$. Among all solutions, we consider a feasible integral b-flow which is of minimum cost. This flow corresponds to an optimal solution of the weighted bipartite matching problem on the graph G.

Applying the Hungarian algorithm

A second approach to solving the weighted bipartite matching problem is the so-called *Hungarian algorithm.*

We extend G to a complete bipartite graph by adding all the edges e to G which are missing, and assign them a weight of $c_e = 0$. This way, all maximum weight matchings in G can be assumed to be of the same size.

If $|U| \neq |V|$, we balance the sizes of U and V by adding an appropriate number of nodes to the smaller of the two sets, and add further edges of weight 0 to the graph. Finally, the underlying graph is the complete bipartite graph $K_{n,n}$, where n is the number of nodes in the larger of the two color classes of G.

Now, for each matching in G which is of maximum weight there is a perfect matching in $K_{n,n}$ which is of the same weight, and vice versa. The Hungarian algorithm, listed as Algorithm B.2 in Appendix B, determines a perfect matching of maximum weight with respect to nonnegative edge weights in a complete bipartite graph $K_{n,n}$. The algorithm is due to Kuhn [35], [36], who also proved its polynomial running-time.

Theorem 1.13. *The weighted bipartite matching problem is polynomially solvable.*

1.3.3. Solving weighted bipartite perfect matching problems

Let $G = (U \mathbin{\dot\cup} V, E)$ be a bipartite graph with $|U| = |V| = n$ and let $c \in \mathbb{R}^E$ be a vector of edge weights. We now consider the task of finding a perfect matching in G which is of maximum weight.

As the weighted bipartite perfect matching problem is a special case of the weighted bipartite matching problem, the problem can be transformed into a minimum cost flow problem as described in Section 1.3.2. As the resulting matching must be a perfect matching, the parameter r is fixed to n. Then a feasible integral b-flow of minimum cost in D corresponds to a an optimal solution to the weighted bipartite perfect matching problem on the graph G.

It is further possible to use the Hungarian algorithm to determine a perfect matching of maximum weight in G. We transform G into the complete bipartite graph $K_{n,n}$ by adding additional 0-weight edges. Now, we ensure that the Hungarian algorithm chooses only those edges in $K_{n,n}$ to be in a matching which have originally been in G.

To this end, we increase the weight c_e for all edges $e \in E$. One should note that adding a constant C to the weights of all edges in G does not change the set of maximum weight perfect matchings in G. This holds as all perfect matchings in G are of the same size n, and thus adding C to the edge weights is equivalent to adding the constant nC to the objective function.

Let $c_{\min} := \min_{e \in E} c_e$ and $c_{\max} := \max_{e \in E} c_e$. Let C be a constant with

$$C > n(c_{\max} - c_{\min}) - c_{\max}.$$

When adding C to the edge weight c_e for all $e \in E$, the resulting edge weights c'_e for all edges e in $K_{n,n}$ are as follows:

$$c'_e = \begin{cases} c_e + C & \text{if } e \in E, \\ 0 & \text{if } e \notin E. \end{cases}$$

Let M be a perfect matching in $K_{n,n}$ with $e \in E$ for all $e \in M$. Further, let N be a perfect matching in $K_{n,n}$ such that there is an edge $e \in N$ with $e \notin E$. The following holds:

$$\begin{aligned} c'(M) &\geq n(c_{\min} + C) \\ &= nc_{\min} + nC + C - C \\ &> nc_{\max} - c_{\max} + nC - C \\ &= (n-1)(c_{\max} + C) \\ &\geq c'(N) \end{aligned}$$

Hence, a perfect matching consisting only of edges which are in E is of higher weight than any perfect matching which contains an edge that is not in E. As the Hungarian algorithm determines a perfect matching of maximum weight, it solves the weighted bipartite matching problem.

Theorem 1.14. *The weighted bipartite perfect matching problem is polynomially solvable.*

1.3.4. Solving matching problems on general graphs

When a matching problem is stated on a nonbipartite graph $G = (V, E)$, it is not possible to transform the problem into a maximum flow problem. This is because the property of a graph to be bipartite is essential for defining an orientation of the edges in E as described in Section 1.3.1.

An algorithm for finding maximum matchings in nonbipartite graphs is presented in Algorithm B.1 in Appendix B. The algorithm is based on so-called alternating trees and a contraction procedure (called the *shrinking* procedure in Algorithm B.1) which is credited to Edmonds [20]. Due to the running-time of the algorithm, we have the following complexity result for the matching problem.

Theorem 1.15. *The matching problem on a graph $G = (V, E)$ is polynomially solvable.*

1.3.5. Solving assignment problems

Let $K_{n,n} = (U \cup V, E)$ be the underlying graph of an assignment problem with corresponding cost vector $c \in \mathbb{R}^E$. The assignment problem is a special case of the weighted bipartite perfect matching problem. Hence, it can be polynomially solved by transforming it into a minimum cost flow problem (see Section 1.1.2) and by using the Hungarian algorithm (see Section 1.3.3).

As the Hungarian algorithm is designed to work on complete bipartite graphs, the underlying graph of the assignment problem does not need to be adapted. So, the only transformation step needed to be done before applying the Hungarian algorithm to an instance of the assignment problem is to define the edge weights.

The task of the assignment problem is to find a perfect matching of minimum cost and the Hungarian algorithm determines a perfect matching of maximum weight. Further, all edge weights in the Hungarian algorithm need to be nonnegative. We define the edge weights for the Hungarian algorithm as $c'_e := -c_e + C$, with $C := \max_{e \in E}\{c_e\}$. Thus, c' is a vector of nonnegative edge weights. Finally, each solution of the Hungarian algorithm of weight z corresponds to a solution of the assignment problem of cost $-z + nC$.

1.4. The perfect matching polytope and the matching polytope

The aim of this section is to provide definitions of matching polytopes together with those properties that are of further interest in this work. Investigations on the structure of polytopes of matchings with a special type of additional side constraint are presented in Chapter 6.

Definition 1.16. *Let $G = (V, E)$ be a graph. Then, the* perfect matching polytope *$P_{\text{perfect matching}}(G)$ is the convex hull of incidence vectors of perfect matchings in G. The* matching polytope *$P_{\text{matching}}(G)$ is the convex hull of incidence vectors of matchings in G.*

Given a graph $G = (V, E)$, both polyhedrons $P_{\text{perfect matching}}(G)$ and $P_{\text{matching}}(G)$ are polytopes in $\mathbb{R}^E$.

1.4.1. Bipartite perfect matching polytopes and bipartite matching polytopes

Let $G = (V, E)$ be a bipartite graph. We consider the perfect matching polytope, first. Each point $x \in P_{\text{perfect matching}}(G)$ satisfies

$$\sum_{e\in\delta(v)} x_e = 1 \qquad \forall\ v \in V, \tag{1.24}$$

$$x_e \geq 0 \qquad \forall\ e \in E. \tag{1.25}$$

Birkhoff [9] showed that constraints (1.24) and (1.25) are not only necessary but also sufficient to describe $P_{\text{perfect matching}}(G)$ (see [49] for Birkhoff's proof in the terminology of perfect matchings).

Theorem 1.17. *Let $G = (V, E)$ be a bipartite graph. Then, the perfect matching polytope $P_{\text{perfect matching}}(G)$ is determined by (1.24), (1.25).*

We will give a reason for Theorem 1.17 in terms of total unimodularity. This allows us to transfer this result to the matching polytope on bipartite graphs. Let $A \in \{0, 1\}^{|V|\times|E|}$ be the node-edge incidence matrix of the graph $G = (V, E)$. Then, the constraints (1.24) can be written as $Ax = \mathbf{1}$. Using the partitioning argument (see [43]) to show total unimodularity, the following holds true:

Theorem 1.18. *Let G be a graph. G is bipartite if and only if its node-edge incidence matrix A is total unimodular.*

Due to Theorem 1.18, the polytope described by (1.24) and (1.25) for the bipartite graph G is integral and hence, these constraints determine the perfect matching polytope $P_{\text{perfect matching}}(G)$.

The total unimodularity of A also implies that the polytope described by the following constraints is integral for the bipartite graph G:

$$\sum_{e\in\delta(v)} x_e \leq 1 \qquad \forall\, v \in V, \tag{1.26}$$

$$x_e \geq 0 \qquad \forall\, e \in E. \tag{1.27}$$

As these constraints are necessary for any matching in G, we get the following result:

Theorem 1.19. *Let $G = (V, E)$ be a bipartite graph. Then, the matching polytope $P_{\text{matching}}(G)$ is determined by (1.26), (1.27).*

Concerning the dimension of the perfect matching polytope of a bipartite graph, Naddef [42] proved the following result:

Theorem 1.20. *Let $G = (V, E)$ be a bipartite graph which contains at least one perfect matching. Let E_0 be the set of edges contained in at least one perfect matching in G and let k be the number of components of the graph (V, E_0). Then,*

$$\dim\left(P_{\text{perfect matching}}(G)\right) = |E_0| - |V| + k.$$

1.4.2. Nonbipartite perfect matching polytopes and nonbipartite matching polytopes

If the graph $G = (V, E)$ is not bipartite, then the constraints (1.24), (1.25) do not determine the perfect matching polytope. To see this, consider the complete graph K_3. The vector $x = \left(\frac{1}{2}, \frac{1}{2}, \frac{1}{2}\right)^\top$ satisfies these constraints but cannot be written as a convex combination of incidence vectors of perfect matchings in K_3, as there are no perfect matchings in K_3. Thus, $x \notin P_{\text{perfect matching}}(G)$.

Edmonds [19] showed that adding constraints of the form

$$x(\delta(U)) \geq 1 \qquad \forall\, U \subseteq V \text{ with } |U| \text{ odd} \tag{1.28}$$

to the set of constraints (1.24), (1.25) suffices to determine the perfect matching polytope.

Theorem 1.21. *Let $G = (V, E)$ be any graph. Then, the perfect matching polytope $P_{\text{perfect matching}}(G)$ is determined by (1.24), (1.25), (1.28)*

Edmonds also offers a characterization of the matching polytope of a general graph $G = (V, E)$:

Theorem 1.22. *Let $G = (V, E)$ be any graph. Then, the matching polytope $P_{\text{matching}}(G)$ is determined by*

$$\begin{aligned} \sum_{e\in\delta(v)} x_e &\leq 1 && \forall\, v \in V, \\ x_e &\geq 0 && \forall\, e \in E, \\ x(E(U)) &\leq \left\lfloor \frac{1}{2}|U| \right\rfloor && \forall\, U \subseteq V \textit{ with } |U| \textit{ odd.} \end{aligned}$$

1.4.3. The polytope of nonbipartite matching problems with an additional cardinality constraint

In this section we consider the polytope of a matching problem with an additional cardinality constraint. This is of particular interest as investigations regarding additional constraints on matchings are the main subject of this work.

In Theorem 1.19 and Theorem 1.22 formulations of the matching polytope $P_{\text{matching}}(G)$ are given for G being a bipartite graph and a nonbipartite graph, respectively. Let $G = (V, E)$ be any graph and let k and l be nonnegative integers with $k \leq l$. We now consider the matching polytope for the case that a matching M in G must satisfy $k \leq |M| \leq l$.

In [49], Schrijver presents a formulation of the convex hull of incidence vectors of matchings whose size is at least k and at most l. The result is presented in the next theorem. This determines the polytope corresponding to the matching problem with an additionally imposed cardinality constraint.

Theorem 1.23. *Let $G = (V, E)$ be a graph and let $k, l \in \mathbb{Z}_0^+$ with $k \leq l$. Then, the convex hull of incidence vectors of matchings M satisfying $k \leq |M| \leq l$ is equal to the set of those vectors x in the matching polytope $P_{\text{matching}}(G)$ satisfying $k \leq \mathbf{1}^\top x \leq l$.*

In other words, the polytope corresponding to the matching problem with a cardinality constraint is determined by adding

$$k \leq \sum_{e \in E} x_e \leq l \tag{1.29}$$

to the formulation of $P_{\text{matching}}(G)$. Schrijver describes this as the fact that "certain slices of the matching polytope are again integer polytopes".

It is essential in Theorem 1.23 that the cardinality constraint affects the entire matching, i.e. the incidence vector components corresponding to all edges in E appear in constraint (1.29). To see this, we consider the special case where $k = l$. Now, the additional cardinality constraint (1.29) requires matchings to be *exactly* of cardinality k (resp. l). If this cardinality constraint applies only to a subset R of the edges, the polytope $P_{R,k} := \{x \in \mathbb{R}_+^E \mid x \in P_{\text{matching}}(G) \text{ and } x(R) = k\}$ is not integral in general, as the following example shows.

Let $G = (V, E) = K_{2,2}$ with $E = \{[i, j] \mid i, j = 1, 2\}$ and let $k = l = 1$. Further, let $R := \{[1, 1], [2, 2]\}$. The vector $x \in \mathbb{R}^E$ with $x_e = \frac{1}{2}$ for all $e \in E$ is in $P_{R,k}$ but it cannot be written as a convex combination of incidence vectors of matchings M in G satisfying $|M \cap R| = k$.

Matching problems with an exact cardinality constraint affecting only a special subset of the edges of a graph are discussed in Section 2.2 and Chapter 4.

1.5. The 3-dimensional matching problem

In Section 1.2 we introduced the perfect matching problem, which is polynomially solvable. In this section we show a more general variant of the perfect matching problem, which is NP-complete. This problem is stated on a generalized concept of graphs, called hypergraphs (see [49]):

Definition 1.24. *Let V be a finite set and let $\mathcal{E}$ be a family of subsets of V. The pair $H = (V, \mathcal{E})$ is called a* hypergraph. *Any element of V is called a* node *and any element of $\mathcal{E}$ is called an* edge.

A graph is a special case of a hypergraph where all edges are of cardinality at most 2. A matching $\mathcal{M}$ in a hypergraph $H = (V, \mathcal{E})$ is a subset of $\mathcal{E}$, represented by an incidence vector $y \in \{0,1\}^{\mathcal{E}}$, which satisfies

$$\sum_{F \in \mathcal{E} : F \ni v} y_F \leq 1 \qquad \forall\, v \in V.$$

Using the terminology of a hypergraph, the *3-dimensional matching problem* is formulated next (compare [31]). Let $H = (U \cup V \cup W, \mathcal{E})$ be a hypergraph with pairwise disjoint color classes U, V and W, each of them of size n, and an edge set $\mathcal{E}$ where each edge is an element of $U \times V \times W$.

Problem Formulation 1.25 (3-Dimensional matching)**.** *The task of the* 3-dimensional matching problem (3DM) *is to determine a matching $\mathcal{M}$ in H which is of size n.*

The following result is due to Karp [31], who polynomially reduces the partitioning problem (see Appendix C) to 3DM.

Theorem 1.26. *The 3-dimensional matching problem is NP-complete.*

The 2-dimensional counterpart of 3DM is the perfect matching problem on a bipartite graph $G = (X \uplus Y, E)$ with $|X| = |Y|$. In classical (2-dimensional) matching, each edge has two end-nodes (assumed that the graph is free of loops) at which it can meet with one or more edges. In 3-dimensional matching, there is a third color class (or third dimension) W and each edge contains 3 nodes at which it can meet with other edges. Due to this additional node in an edge, each edge appears in 3 inequalities of the matching constraints, rather than in only 2 as it is the case in the classical perfect matching problem. Thus, the matching condition that no two edges are allowed to meet at any node is more extensive in 3-dimensional matching.

2. Resource Constrained Matching

There are two classes of matching problems with additional side constraints which we summarize as *resource constrained matching problems*: The couple constrained matching problem and the level constrained matching problem. While both problems are matching problems, they differ in the type of side constraints additionally imposed.

Couple constrained matching problems demand in each additional side constraint that for a given pair of edges both edges have the same matching activity. Level constrained matching problems demand in one additional side constraint that an exact number of matching edges are elements of a special set called the set of on-level edges.

Couple constrained matching problems vary in the number of additional side constraints, with all of them having a support of size 2. Level constrained matching problems have just one side constraint, whose support can grow with the problem size. Hence, the couple constrained matching problem and the level constrained matching problem represent two classes of matching problems with additional side constraints of different characteristics. These are the main problems considered in this work.

The aim of Sections 2.1 and 2.2 is to introduce these problems. In each section we first give the mathematical formulation of the considered problem and show which other matching variants with the same type of additional side constraints are equivalent. We then define the assignment problem with the same set of additional side constraints.

In Section 2.3, we take a closer look at the additional side constraints of resource constrained matching problems. We show that if the number of couples or the number of demanded on-level edges is fixed, then the couple constrained matching problem and the level constrained matching problem can be solved polynomially.

We conclude this chapter with a section addressing other matching problems from the literature which have additional side constraints that are related to those in the couple constrained matching problem and the level constrained matching problem. We also investigate the relation of resource constrained matching problems to symmetric matching problems. Furthermore, the integer equal flow problem – to which the couple constrained matching problem and the level constrained matching problem both can be polynomially reduced – is presented.

2.1. The couple constrained matching problem

2.1.1. Introduction and problem formulation

The first problem we introduce is the couple constrained matching problem. In this problem it is required that for given pairs of edges each of the two edges which appear in the same pair must have the same matching activity. These types of constraints are called *couple constraints*.

Couple constraints have practical applications especially when there are pairs of node-to-node assignments that are mutually dependent. This may appear, for instance, in a workers to jobs matching problem, where there are pairs of workers that build a team and there are pairs of jobs which require to be worked on by members of the same team.

Couple constraints can also be used to include conditions in modeling which become relevant after an assignment has happened. An example of this is assigning jobs to machines where processing a job means producing an end-product. Whenever there are two end-products which are known to be further processed, it is preferable to assign the two preceding jobs to machines which offer an advantage for the next processing step. This advantage could be a spatial proximity of the machines or the fact that both machines guarantee the same production quality. For example, when assigning the tasks of producing different car parts to production plants, it is desirable to produce two parts in the same plant if they appear together in a further production step. This diminishes transportation costs between plants.

Further practical applications are given by Aboudi et al. in [1]. They consider the problem of assigning courses to classrooms, where two consecutive courses are to be held in the same classroom in two consecutive time slots. They also consider job to worker assignments where it is necessary to take into consideration that in married couples both partners must be assigned jobs that are located in the same city. In [10], stability issues of a similar problem, namely the problem of matching married couples to a pair of positions at hospitals, are considered. Padberg and Sassano [45] investigate a more general version of the couple constrained matching problem, which they call matching with bonds. This is a maximum weight matching problem, where bonds replace the couples in the additional side constraints. Bonds are sets of edges of arbitrary size. As a part of their analysis, they also consider the case of all bonds having a maximum cardinality of 2, so that the problem becomes a maximum weight couple constrained matching problem.

The couple constrained matching problem we define in this section is a maximum cardinality matching problem with a varying number of specific additional side constraints. Each side constraint ties two variables x_e and x_f in the simplest way, i.e. $x_e = x_f$. This type of side constraint requires that the matching either includes both edges e and f or excludes them both. Each such constraint is of a simple structure, but their number is not limited in advance.

Definition 2.1. *Let $G = (V, E)$ be a graph with $V = \{v_1, \dots, v_n\}$. A* couple collection *$\mathcal{F} = \{F_1, \dots, F_k\}$ in G is defined to be a set of pairwise disjoint pairs of edges in E, i.e. $F_i \cap F_j = \emptyset$ for all $F_i, F_j \in \mathcal{F}$ with $i \neq j$. A pair $F = \{e, f\} \in \mathcal{F}$ is referred to as a* couple*.*

Given a graph G and a couple collection $\mathcal{F}$ in G as problem input, we now formulate the couple constrained matching problem:

Problem Formulation 2.2 (Couple constrained matching)**.** *The* couple constrained matching problem (CCMP) *is stated as the following integer linear program:*

$$\begin{aligned}
\max \quad & \sum_{e \in E} x_e && && (2.1)\\
s.t. \quad & \sum_{e \in \delta(v_i)} x_e \leq 1 && \forall\ i = 1, \dots, n && (2.2)\\
& x_e = x_f && \forall\ e, f \in F, \quad \forall\ F \in \mathcal{F} && (2.3)\\
& x_e \in \{0, 1\} && \forall\ e \in E. && (2.4)
\end{aligned}$$

Problem ((2.1), (2.2), (2.4)) is a maximum cardinality matching problem on a graph $G = (V, E)$. The additional constraints (2.3) are the crucial ones for this problem. They demand that either both edges of a couple $F \in \mathcal{F}$ are selected to be in the matching or none of them. These side constraints will be called *couple constraints*. Clearly, if $\mathcal{F}$ does not contain any couple, then we have the classical matching problem. When a matching in G satisfies the additional constraints (2.3), we refer to it as a *solution matching* of the corresponding CCMP instance.

Without loss of generality, for each couple $F = \{e, f\}$ it holds that e and f are not incident to the same node, as otherwise at most one of these edges could possibly be in a matching and consequently, due to the couple constraints, none of them would be in a matching. Therefore, we can delete such e and f from the edge set E in advance.

2.1.2. Polynomially equivalent optimization variants

The couple constrained matching problem is defined as a maximum cardinality problem. With regards to complexity, it does not make a difference whether it is formulated as a maximum cardinality matching problem or as a perfect matching problem. We summarize the complexity relationship between these two problem variants in Theorem 2.4. For that, we define the following problem variant of the CCMP:

Problem Formulation 2.3 (Couple constrained perfect matching)**.** *The* couple constrained perfect matching problem (CCPMP) *is the problem of finding a perfect matching satisfying the couple constraints in a graph.*

Theorem 2.4. *The following problems are polynomially equivalent:*

- *The couple constrained matching problem.*
- *The couple constrained perfect matching problem.*

Proof. It is clear that the CCPMP can be polynomially reduced to the CCMP on the same graph. We now show the opposite direction. For that, we consider the CCMP as a decision problem. Let $(G, \mathcal{F}, k)$ be a problem instance of the decision version of the CCMP with $G = (V, E)$ being a graph, $\mathcal{F}$ being a couple collection on G and k being a nonnegative integer. The question is whether there exists a matching in G which fulfills the couple constraints concerning $\mathcal{F}$ and which is of size at least k. We now show how $(G, \mathcal{F}, k)$ is polynomially reduced to the problem instance $(G', \mathcal{F}')$ of the CCPMP.

Let $n := |V|$ and let $d := n - 2k$. The graph $G' = \big(V \cup V', E \cup E'\big)$ is defined by

$$\begin{aligned} V' &:= \{v'_1, \dots, v'_d\}, \\ E' &:= \big\{[v, v'_i] \mid v \in V, i = 1, \dots, d\big\} \cup \big\{[v'_i, v'_j] \mid i, j = 1, \dots, d; i \neq j\big\}. \end{aligned}$$

The couple collection $\mathcal{F}'$ contains the same couples as $\mathcal{F}$, thus $\mathcal{F}' := \mathcal{F}$.

If G' contains a perfect matching, then at least $n - d = 2k$ nodes in V are covered by edges in E. Hence, if G' contains a perfect matching satisfying the couple constraints concerning $\mathcal{F}'$, then there is a matching of size at least k in G, which fulfills the couple constraints concerning $\mathcal{F}$. On the other hand, each matching in G which is of size at least k can be extended to a perfect matching in G'. Hence, if G contains a matching of size at least k which fulfills the couple constraints concerning $\mathcal{F}$, then G' contains a perfect matching satisfying the couple constraints concerning $\mathcal{F}'$. Obviously, the reduction is polynomial in the size of G. $\square$

The polynomial equivalence of the CCMP and the CCPMP also holds when restricting the underlying graphs to be bipartite. Further, on bipartite graphs one can formulate the following problem variant of the CCMP:

Problem Formulation 2.5 (Couple constrained complete matching)**.** *The* couple constrained complete matching problem (CCCMP) *is the problem of finding a complete matching satisfying the couple constraints in a bipartite graph.*

Theorem 2.6. *The following problems are polynomially equivalent:*

- *The couple constrained matching problem on a bipartite graph.*
- *The couple constrained complete matching problem.*
- *The couple constrained perfect matching problem on a bipartite graph.*

Proof. It is easy to see that the CCPMP on bipartite graphs can be polynomially reduced to the CCCMP, and that the CCCMP can be polynomially reduced to the CCMP on bipartite graphs. In these reductions, the underlying graph and couple collection do not change.

We now show the opposite direction, and start with a reduction from the CCMP on bipartite graphs to the CCCMP. Let $(G, \mathcal{F}, k)$ be a problem instance of the decision version of the CCMP, with $G = (X \mathbin{\dot\cup} Y, E)$ being a bipartite graph, $\mathcal{F}$ being a couple collection on G and k being a nonnegative integer. The question is whether there exists a matching in G of size at least k, which fulfills the couple constraints corresponding to $\mathcal{F}$. Without loss of generality, let X be the smaller of the two color classes in G and let $n := |X|$. We now show how $(G, \mathcal{F}, k)$ is polynomially reduced to the problem instance $(G', \mathcal{F}')$ of the CCCMP. The bipartite graph $G' = \big(X \mathbin{\dot\cup} (Y \cup Y'), E \cup E'\big)$ is defined by

$$\begin{aligned} Y' &:= \{v_1, \dots, v_{n-k}\}, \\ E' &:= \{[x, v_i] \mid x \in X, i = 1, \dots, n-k\}. \end{aligned}$$

The couple collection $\mathcal{F}'$ contains the same couples as $\mathcal{F}$, thus $\mathcal{F}' := \mathcal{F}$. Now, G' contains a matching covering X and fulfills the couple constraints concerning $\mathcal{F}'$ if and only if G contains a matching of size at least k which fulfills the couple constraints concerning $\mathcal{F}$.

Next, we show a reduction from the CCCMP to the CCPMP on bipartite graphs. Let $(G, \mathcal{F})$ be a problem instance of the CCCMP, with $G = (X \mathbin{\dot\cup} Y, E)$ being a bipartite graph and $\mathcal{F}$ being a couple collection on G. Again, without loss of generality, let X be the smaller of the two color classes in G and let $d := |Y| - |X|$ denote the difference in the cardinality of the two color classes. We now show how $(G, \mathcal{F})$ is polynomially reduced to the problem instance $(G', \mathcal{F}')$ of the CCPMP. The bipartite graph $G' = \big((X \cup X') \mathbin{\dot\cup} Y, E \cup E'\big)$ is defined by

$$\begin{aligned} X' &:= \{v_1, \dots, v_d\}, \\ E' &:= \{[v_i, y] \mid y \in Y, i = 1, \dots, d\}. \end{aligned}$$

Due to the nodes in X', the two color classes in G' are of same cardinality. As before, the couple collection does not change along the reduction, i.e. $\mathcal{F}' := \mathcal{F}$. Now, G' contains a perfect matching which fulfills the couple constraints concerning $\mathcal{F}'$ if and only if G contains a matching which covers all nodes in X and fulfills the couple constraints concerning $\mathcal{F}$.

It is easy to see that all reductions described here are polynomial in the input size of the corresponding problem. □

2.1.3. The couple constrained assignment problem

When couple constraints are imposed on the classical assignment problem, we get the assignment problem with additional constraints demanding that in specified pairs either both of the edges or neither of the edges are in the assignment.

Similar to a couple collection, let $\mathcal{F} = \{F_1, \dots, F_k\}$ be a set containing pairwise disjoint sets of the form $F_h = \{(i, j), (k, l)\}$ with $i, j, k, l \in \{1, \dots, n\}$. Further, let $c_{ij} \in \mathbb{R}$ be the cost of assigning row i to column j for all $i, j = 1, \dots, n$.

Problem Formulation 2.7 (Couple constrained assignment)**.** *The* couple constrained assignment problem (CCAP) *is stated as the following integer linear program:*

$$\begin{aligned} \min \quad & \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} && && (2.5) \\ s.t. \quad & \sum_{j=1}^{n} x_{ij} = 1 && \forall\ i = 1, \dots, n && (2.6) \\ & \sum_{i=1}^{n} x_{ij} = 1 && \forall\ j = 1, \dots, n && (2.7) \\ & x_{ij} = x_{kl} && \forall\ (i, j), (k, l) \in F, \quad \forall\ F \in \mathcal{F} && (2.8) \\ & x_{ij} \in \{0, 1\} && \forall\ i, j = 1, \dots, n. && (2.9) \end{aligned}$$

This problem can also be viewed as a minimum cost couple constrained perfect matching problem on the complete bipartite graph $K_{n,n}$. This interpretation allows us to refer to edges and their costs in the underlying complete bipartite graph of an assignment problem.

2.2. The level constrained matching problem

2.2.1. Introduction and problem formulation

In this section we introduce the level constrained matching problem. It is a resource constrained matching problem with a single additional side constraint. This constraint demands a feasible matching to contain an exact number of so-called *on-level edges* in a bipartite graph. Given fixed indices of the nodes in the graph, these are the edges with end-nodes that have the same index.

Restrictions on the number of selected edges from a given subset R of the edge set are considered in the literature e.g. by Karzanov [32], Yi et. al. [52] and Alfakih et. al [6], [7]. A common formulation of this problem in the literature is a generalized formulation as a perfect matching problem where an additional linear constraint $\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} = k$ is required to be fulfilled. As this problem is NP-hard in general (see [13]), the coefficients c_{ij} are restricted to be 0-1-valued. In this particular case, those edges whose corresponding variables have coefficient 1 construct the set R. Instead of defining the additional side constraint by means of a set $R \subseteq E$, also the terminology of edges being colored blue and red is used in the literature. Then, the problem is stated as the problem of finding a perfect matching which contains exactly k red edges.

We choose the notation which refers to a subset $R \subseteq E$. For the problem we introduce in this section, we have that R is the set of on-level edges in G.

Definition 2.8. *Let $G = (U \cup V, E)$ be a bipartite graph with $U = \{u_1, \dots, u_n\}$, $V = \{v_1, \dots, v_n\}$. Furthermore, let the assignment of indices to nodes in U and V be fixed. The set of edges E is partitioned into two subsets. An edge of the form $[u_i, v_i]$ for $i = 1, \dots, n$ will be called an* on-level edge*. All other edges, i.e. those of the form $[u_i, v_j]$ for $i, j = 1, \dots, n$ with $i \neq j$ will be called* off-level edges*. The graph G is called a* level graph *if $[u_i, v_i] \in E$ for all $i = 1, \dots, n$.*

Next, we formulate the problem of finding a maximum matching with an exact number of on-level edges. Let $G = (U \cup V, E)$ be a bipartite graph with $U = \{u_1, \dots, u_n\}$, $V = \{v_1, \dots, v_n\}$ and let k be an integer with $0 \leq k \leq n$. As in Definition 2.8, let the assignment of indices to nodes in U and V be fixed, so that the set of on-level edges in G is uniquely defined.

To add clarity, we emphasize that both the graph G (with fixed node indices) and the parameter k are part of the problem input. We now formulate the level constrained matching problem:

Problem Formulation 2.9 (Level constrained matching)**.** *The* level constrained matching problem (LCMP) *is stated as the following integer linear program:*

$$\begin{aligned}
\max \quad & \sum_{e \in E} x_e && && (2.10)\\
s.t. \quad & \sum_{e \in \delta(u_i)} x_e \leq 1 && \forall\, i = 1, \dots, n && (2.11)\\
& \sum_{e \in \delta(v_i)} x_e \leq 1 && \forall\, i = 1, \dots, n && (2.12)\\
& \sum_{[u_i, v_i] \in E} x_{[u_i, v_i]} = k && && (2.13)\\
& x_e \in \{0, 1\} && \forall\, e \in E. && (2.14)
\end{aligned}$$

Problem ((2.10), (2.11), (2.12), (2.14)) is a maximum cardinality matching problem on the bipartite graph G. The additional constraint (2.13) affects the number of on-level edges in a feasible matching. It demands that exactly k of them are contained in a matching. This side constraint will be called the *level constraint.* When a matching in G satisfies the additional constraint (2.13), we refer to it as a solution matching of the corresponding LCMP instance.

Annotation. *An instance of the LCMP depends on the indexation of the nodes in the graph $G = (U, V)$. Throughout this work we assume that for each instance of LCMP (and for other problems which contain a level constraint) such a indexation is given as part of the problem input, without mentioning it explicitly. The only type of renumbering of nodes under which an LCMP instance is invariant is where the nodes in U and V are renumbered analogously, i.e. $u_i \in U$ is renamed $u_j \in U$ if and only if $v_i \in V$ is renamed $v_j \in V$.*

2.2.2. Polynomially equivalent optimization variants

The underlying matching problem in the LCMP is a maximum cardinality matching problem. We will show that concerning its complexity it makes no difference whether the underlying matching problem is a maximum cardinality, complete or perfect matching problem. We emphasize that the level constrained complete matching problem is the only variant of the LCMP where the two color classes of the underlying bipartite graph are allowed to be of different cardinality.

Problem Formulation 2.10 (Level constrained complete matching)**.** *The* level constrained complete matching problem (LCCMP) *is the problem of finding a complete matching satisfying the level constraint in a bipartite graph.*

Problem Formulation 2.11 (Level constrained perfect matching)**.** *The* level constrained perfect matching problem (LCPMP) *is the problem of finding a perfect matching satisfying the level constraint in a bipartite graph.*

Theorem 2.12. *The following problems are polynomially equivalent:*

- *The level constrained matching problem.*
- *The level constrained complete matching problem.*
- *The level constrained perfect matching problem.*

The proof is similar to that of Theorem 2.6, where couple constrained matching problems on bipartite graphs are considered. Here, we will take care that no additional on-level edges appear in a graph along a problem reduction, as the level constraint always refers to the entire set of on-level edges.

Proof. It is easy to see that the LCPMP can be polynomially reduced to the LCCMP, and that the LCCMP can be polynomially reduced to the LCMP. In these reductions, the underlying graph and the parameter of the level constraint do not change.

We now show the opposite direction, starting with a reduction from the LCMP to the LCCMP. Let (G, k, l) be a problem instance of the decision version of the LCMP, where $G = (X \uplus Y, E)$ is a bipartite graph with $X = \{x_1, \ldots, x_{n_1}\}$ and $Y = \{y_1, \ldots, y_{n_2}\}$. The parameters k and l are nonnegative integers serving as parameter for the level constraint and as decision parameter, respectively. The question is whether there exists a matching in G which contains exactly k on-level edges and which is of size at least l.

Without loss of generality, let $n_1 \leq n_2$. Further, let $d := n_1 - l$. If $l > n_1$ then there is no matching in G which is of cardinality at least l. We now show how (G, k, l) is polynomially reduced to the problem instance (G', k') of the LCCMP. The bipartite graph $G' = \big(X \uplus (Y \cup Y'), E \cup E'\big)$ is defined by

$$\begin{aligned} Y' &:= \{y_{n_2+1}, \ldots, y_{n_2+d}\}, \\ E' &:= \{[x, y_{n_2+i}] \mid x \in X, i = 1, \ldots, d\}. \end{aligned}$$

As G' contains the same on-level edges as G, we define $k' := k$. Obviously, there is a matching in G' which contains exactly k' on-level edges and which covers X, if and only if there is a matching in G which contains exactly k on-level edges and which is of size at least l.

Next, we show a reduction from the LCCMP to the LCPMP. Let (G, k) be a problem instance of the LCCMP, where $G = (X \uplus Y, E)$ is a bipartite graph with $X = \{x_1, \ldots, x_{n_1}\}$, $Y = \{y_1, \ldots, y_{n_2}\}$ and where k is a nonnegative integer. Again, without loss of generality, let $n_1 \leq n_2$ and let $d := n_2 - n_1$ denote the difference in the cardinality of the two color classes. In case $d = 1$, the LCCMP instance can be solved by solving n_2 problem instances of the LCPMP on the graphs $G - y_1, \ldots, G - y_{n_2}$, with unchanged parameter k. One should note that the number of LCPMP instances to solve is polynomially bounded by the size of G. In the case where $d \geq 2$, we polynomially reduce (G, k) to the problem instance (G', k') of the LCPMP. The bipartite graph $G' = \big((X \cup X') \uplus Y, E \cup E'\big)$ is defined by

$$\begin{aligned} X' &:= \{x_{n_1+1}, \ldots, x_{n_1+d}\}, \\ E' &:= \{[x_{n_1+i}, y_j] \mid i = 1, \ldots, d; j = 1, \ldots, n_2 \text{ with } n_1 + i \neq j\}. \end{aligned}$$

The edges in E' connect all nodes in X' with all nodes in Y while leaving out all on-level edges. Due to the nodes in X', the two color classes in G' are of same cardinality. As before, the parameter for the level constraint does not change, i.e. $k' := k$. Now, G' contains a perfect matching which contains exactly k' on-level edges if and only if G contains a matching which covers all nodes in X and contains exactly k on-level edges.

Finally, all reductions used in this proof are polynomial in the size of their input problems. □

Theorem 2.12 allows us to switch between different variants of underlying matching problems when considering complexity issues of level constrained matching problems. It is clear that a complexity result for any of the three variants also applies for the other two variants.

2.2.3. The level constrained assignment problem

Next, we give a formulation of the assignment problem having a level constraint as additional side constraint. The assignment problem is interpreted in this case as a minimum weight matching problem on a complete bipartite graph $K_{n,n}$. The level constraint requires that the number of on-level edges in a feasible assignment equals a given value.

Let $c_{ij} \in \mathbb{R}$ be the cost of edge $[i, j]$ for all $i, j = 1, \ldots, n$. Further, let k be the parameter of the level constraint, with $0 \leq k \leq n$.

Problem Formulation 2.13 (Level constrained assignment)**.** *The* level constrained assignment problem (LCAP) *is an assignment problem with an imposed level constraint. Its formulation as an integer linear program is as follows:*

$$\begin{array}{lll}
\min & \displaystyle\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} & \\
s.t. & \displaystyle\sum_{j=1}^{n} x_{ij} = 1 & \forall\ i = 1, \ldots, n \\
& \displaystyle\sum_{i=1}^{n} x_{ij} = 1 & \forall\ j = 1, \ldots, n \\
& \displaystyle\sum_{i=1}^{n} x_{ii} = k & \\
& x_{ij} \in \{0, 1\} & \forall\ i, j = 1, \ldots, n.
\end{array}$$

2.3. Input parameters of resource constrained matching problems and complexity

In this section, we will investigate the number of couples occurring in a couple collection in a CCMP instance and the parameter k of the level constraint in an LCMP instance.

We will show that if these parameters are fixed, then the problems can be solved polynomially. The reason for this will be given in the more general context of combinatorial optimization problems rather than referring to the special case of matching problems.

Let E be a finite set and let $\mathcal{L}$ be a subset of the power set 2^E. Further, let $c : E \to \mathbb{R}$ be a weight function on the elements in E. For any set $S \subseteq E$ we denote its weight by $c(S) := \sum_{e \in S} c(e)$. A triple $(E, \mathcal{L}, c)$ serves as problem input for a combinatorial optimization problem.

Definition 2.14. *A* combinatorial optimization problem (COP) *is defined to be the problem*

$$\max\{c(S) \mid S \in \mathcal{L}\}.$$

In order to be compliant with the other problem formulations in this work, we apply the following terminology to COPs. We say that the problem Π is a COP if Π is a – possibly infinite – set of instances $(E, \mathcal{L}, c)$.

When the weighted matching problem or the perfect matching problem are expressed in terms of a COP, each of their instances consists of the following components. The set E corresponds to the set of edges of the underlying graph G. In the case of a weighted matching problem, the set $\mathcal{L}$ consists of all matchings in G and the function c maps all elements in E to their weights. In the case of a perfect matching problem, $\mathcal{L}$ is the set of all perfect matchings in G and c maps all elements in E to any constant.

We now adapt the couple constraints and the level constraint to the more general setting of a COP. For that, let Π be any COP.

We start with imposing couple constraints to Π. To this end, we extend each problem instance $(E, \mathcal{L}, c)$ of Π by a couple collection $\mathcal{F}$. A couple collection on the set E of an instance of a COP is considered to be a set of pairwise disjoint couples, where each couple is a subset of E with cardinality 2. A couple collection for a COP does not differ much from its original definition, except that the elements in the couples are not edges of a graph but rather general elements from the set E.

Definition 2.15. *Let Π be a COP. The problem Π^{cpl} is defined to consist of the problem instances of Π which are extended as follows: For each problem instance $(E, \mathcal{L}, c)$ of Π there is a couple collection $\mathcal{F}$ on E. This yields the problem instance $(E, \mathcal{L}, c, \mathcal{F})$ of Π^{cpl} which is defined to be the problem*

$$\max\{c(S) \mid S \in \mathcal{L} \text{ and } e \in S \Leftrightarrow f \in S \text{ for all } \{e, f\} \in \mathcal{F}\}.$$

We now adapt the level constraint to the context of a COP. As the definition of an on-level edge cannot be applied to a general COP, we replace the level constraint by an equality constraint on an arbitrary subset of E.

Definition 2.16. *Let Π be a COP. The problem Π^{equ} is defined to consist of the problem instances of Π which are extended as follows: For each problem instance $(E, \mathcal{L}, c)$ of Π there is a subset D of E and a nonnegative integer k. This yields the problem instance $(E, \mathcal{L}, c, D, k)$ of Π^{equ} which is defined to be the problem*

$$\max\{c(S) \mid S \in \mathcal{L} \text{ and } |S \cap D| = k\}.$$

One should note that both problems Π^{cpl} and Π^{equ} themselves are COPs again, when in each of their problem instances the additional restrictions are taken as necessary for a set to be in $\mathcal{L}$. Before we establish complexity results for Π^{cpl} with a fixed number of couples and for Π^{equ} with a fixed parameter k, we introduce the following generalization of a COP.

Definition 2.17. *Let Π be a COP. The problem Π^{subset} is defined to consist of the problem instances of Π which are extended as follows: For each problem instance $(E, \mathcal{L}, c)$ of Π there are two arbitrary subsets T and R of E such that $T \subseteq R \subseteq E$. This yields the problem instance $(E, \mathcal{L}, c, T, R)$ of Π^{subset} which is defined to be the problem*

$$\max\{c(S) \mid S \in \mathcal{L} \text{ and } S \cap R = T\}.$$

In this problem, all feasible solutions contain all elements in T but no other elements in R. With the help of the problem Π^{subset} we can give sufficient conditions for a COP with additional side constraints as in the CCMP or the LCMP to be polynomially solvable:

Lemma 2.18. *Let Π be a combinatorial optimization problem such that Π^{subset} is polynomially solvable. Then, the following holds true:*

If the number of couples in each instance of Π^{cpl} is bounded above by a fixed integer p, then Π^{cpl} is polynomially solvable. If the parameter k in each instance of Π^{equ} is fixed, then Π^{equ} is polynomially solvable.

Proof. We begin with the couple constrained problem. Let $(E, \mathcal{L}, c, \mathcal{F})$ be an instance of the problem Π^{cpl}. Without loss of generality, we assume that $\mathcal{F}$ is a couple collection consisting of exactly p couples. For each couple $F_i = \{e_i, f_i\} \in \mathcal{F}$ with $e_i, f_i \in E$, the couple constraints demand that either both elements e_i and f_i are in a solution of the corresponding problem instance, or none of them.

In order to build a problem instance of Π^{subset} let $R := \{e \in E \mid \exists\, F \in \mathcal{F} : e \in F\}$. Taking into account that $\mathcal{F}$ consists of p pairwise disjoint couples, there are 2^p different subsets $T_i \subseteq R$ which correspond to edges in couples that satisfy the couple constraints. Now, we can solve the problem Π^{cpl} on the instance $(E, \mathcal{L}, c, \mathcal{F})$ by solving the problem Π^{subset} on the instances $(E, \mathcal{L}, c, T_i, R)$ for all $i = 1, \dots, 2^p$ and choosing the solution which has the biggest objective value.

Since Π^{subset} is polynomially solvable, each instance $(E, \mathcal{L}, c, T_i, R)$ can be solved in polynomial time. As p is fixed, 2^p is a constant. Hence, solving the problem Π^{subset} on the instances $(E, \mathcal{L}, c, T_1, R), \dots, (E, \mathcal{L}, c, T_{2^p}, R)$ and finding the solution with biggest objective value can be done in polynomial time.

Now, we come to the level constrained problem. Let $(E, \mathcal{L}, c, D, k)$ be an instance of the problem Π^{equ}, where D is an arbitrary subset of E. We denote the cardinality of D by d, and the cardinality of E by m. There are $\binom{d}{k}$ possibilities to choose exactly k elements from the set D. Let $T_i \subseteq D, i = 1, \dots, \binom{d}{k}$, be all subsets of D which are of cardinality k. We define $R := D$. An optimal solution of the instance $(E, \mathcal{L}, c, D, k)$ exists if and only if it also is an optimal solution of the problem Π^{subset} on at least one of the instances $(E, \mathcal{L}, c, T_i, R)$, for $i = 1, \dots, \binom{d}{k}$.

Concerning the number of sets T_i, it holds that

$$\binom{d}{k} = \frac{d!}{(d-k)!k!} \leq d^k \leq m^k.$$

As k is fixed, m^k is a polynomial in m. Together with the assumption that the problem Π^{subset} can be solved in polynomial time this yields that the instance $(E, \mathcal{L}, c, D, k)$ of the problem Π^{equ} can be solved in polynomial time. □

Lemma 2.18 has direct consequences for the CCMP with a fixed number of couples and the LCMP with a fixed cardinality parameter. To see this, we show that when the COP Π in Lemma 2.18 is the matching problem, then the conditions of the lemma are fulfilled.

Theorem 2.19. *The couple constrained matching problem with a fixed number of couples and the level constrained matching problem with a fixed cardinality parameter are polynomially solvable.*

Proof. Let Π be the matching problem and let the graph $G = (V, E)$ be an arbitrary problem instance of the matching problem. Further, let R be an arbitrary subset of E and let T be an arbitrary subset of R. Here, an instance of Π^{subset} on the graph G and the sets T and R is the problem of finding a maximum matching in G which contains all edges in T and none in $R \setminus T$. We now show that this problem can be solved polynomially. For that, we can assume that T itself is a matching, as otherwise it follows directly that there is no matching in G containing all edges in T.

Let $G' = G[V \setminus V(T)] - R$ be the subgraph of G without all edges in R and where the end-nodes (including their incident edges) of all edges in T are deleted, see Figure 2.1. The graph G has a matching M of size s with $M \cap R = T$, if and only if the subgraph G' has a matching M' of size $s - |T|$. Hence, the task is to find a maximum matching M' in G', which can be done in polynomial time (see [20]).

As a consequence, Lemma 2.18 can be applied for the CCMP with a fixed number of couples and for the LCMP with a fixed cardinality parameter. □

2.4. Related problems

This section deals with problems closely related to the CCMP and the LCMP. We first show that the CCMP and the LCMP both are special cases of the maximum integer equal flow problem. We then investigate the special cases of the CCMP and the LCMP where the problems are stated on symmetric bipartite graphs and the solution matchings are demanded to be symmetric. We finish this section with presenting matching problems in the literature which have further variations of additional side constraints.

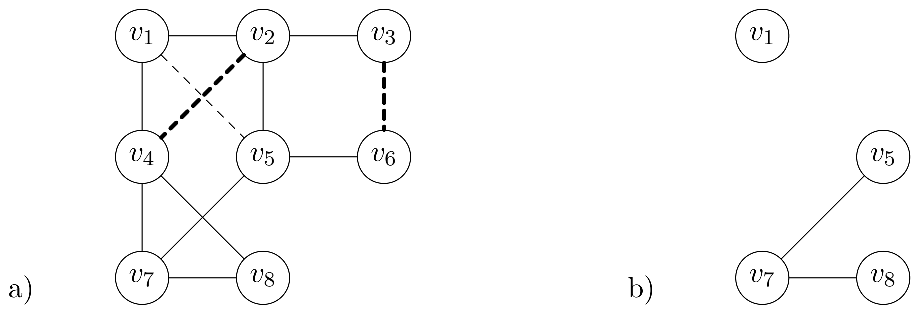


Figure 2.1.: a) Example of a graph $G = (V, E)$, a set $R \subseteq E$ (dashed edges), and a set $T \subseteq R$ (bold edges). A maximum matching M in G with $M \cap R = T$ is of size 3. b) The graph $G' = G[V \setminus V(T)] - R$. A maximum matching M' in G' is of size 1.

2.4.1. Integer equal flows

In this section we introduce the concept of integer equal flows and show that the CCMP and LCMP are special cases of it. The maximum integer equal flow problem is a generalization of the maximum integer flow problem (compare Section 1.1), where it is additionally required that for given arc subsets $R_1, \dots, R_k$ all arcs in the same set carry the same amount of flow. We will show that the maximum integer equal flow problem generalizes the CCMP on bipartite graphs and the LCMP. This is analogous to the classical maximum flow problem which generalizes the classical matching problem on bipartite graphs.

Equal flows (and also integer equal flows) were first introduced by Sahni [48]. Formally, let $D = (V, A)$ be a digraph with $n := |V|$. Let $s, t \in V$ be two distinct nodes denoting the source node and the sink node in D, respectively. Further, let u_{ij} be the nonnegative, integer-valued capacity of arc (i, j) for all $(i, j) \in A$. Now, let $R_1, \dots, R_k$ be pairwise disjoint subsets of A, i.e. each arc in A appears in at most one set R_i for any $i = 1, \dots, k$.

Problem Formulation 2.20 (Maximum integer equal flow)**.** *The* maximum integer equal flow problem *is stated as the following integer linear program:*

$$\max \quad v \tag{2.15}$$

$$\text{s.t.} \quad \sum_{j:(i,j)\in A} x_{ij} - \sum_{j:(j,i)\in A} x_{ji} = \begin{cases} v & \text{if } i = s, \\ 0 & \text{if } i \in V \setminus \{s,t\}, \\ -v & \text{if } i = t \end{cases} \tag{2.16}$$

$$0 \le x_{ij} \le u_{ij} \qquad \forall\ (i,j) \in A \tag{2.17}$$

$$x_{i_1 j_1} = x_{i_2 j_2} \qquad \forall\ (i_1, j_1), (i_2, j_2) \in R_h, \forall\ h = 1, \dots, k \tag{2.18}$$

$$x_{ij} \in \mathbb{Z} \qquad \forall\ (i,j) \in A. \tag{2.19}$$

Problem ((2.15) – (2.17)) describes the classical maximum flow problem. Constraints (2.18) demand that all arcs which appear in the same set R_h, for any $h = 1, \dots, k$, must carry the same amount of flow. We call the sets $R_1, \ldots, R_k$ *homologous sets* and call a constraint of the form (2.18) a *homologous constraint.*

Applications of this problem can be found in [5], [8] and [41]. Among them is the modeling of a water resource system (see [5]). In the corresponding network, water sources are represented by supply nodes and water consuming elements are represented by demand nodes. Water bearing elements, such as rivers and pipes, are represented by arcs. In order to consider several time steps, the underlying network is time-expanded, i.e. it is replicated for each time period. The homologous constraints are then used for modeling the requirement that the amount of potable water is the same for each time period.

The maximum integer equal flow problem is NP-hard, which was proven by Sahni [48]. In [41], Meyers and Schulz investigate the approximability of this problem. They show that the integrality gap between an optimal solution of the LP-relaxation and an optimal integer solution can be arbitrarily large. Concerning its approximability they achieve the following result:

Theorem 2.21. *There is no $2^{n(1-\epsilon)}$-approximation algorithm for the maximum integer equal flow problem for any fixed $\epsilon > 0$, even if a nontrivial solution is guaranteed to exist, unless $P = NP$.*

In the same article, they show that this result holds true even if all homologous arc sets have cardinality 2. This special case is called the maximum paired integer equal flow problem.

Problem Formulation 2.22 (Maximum paired integer equal flow)**.** *The* maximum paired integer equal flow problem *corresponds to the maximum integer equal flow problem where all homologous arc sets are of size 2.*

Our aim is to transform matching problems with additional couple or level constraints into integer equal flow problems. The resulting equal flow problems are maximum paired integer equal flow problems where the capacity of each arc is 1. We first show that even under this capacity restriction, the problem is NP-hard. To this end, we combine a complexity result from Srinathan et al. [50] with a trick used by Meyers and Schulz in [41]. This trick allows us to replace homologous sets of arbitrary size by homologous sets of size 2, as long as the original sets have the property that all arcs in any set R_i emanate from the same node.

Theorem 2.23. *The maximum paired integer equal flow problem with each arc having capacity 1 is NP-hard.*

Proof. Srinathan et al. [50] show that the maximum integer equal flow problem is NP-hard, even if all arcs have capacity 1 and all arcs in a homologous set emanate from the same node. Let $(D = (V, A), s, t, R_1, \ldots, R_k)$ be a problem instance of this problem. Now, we show how to replace all homologous sets of size greater than 2 with homologous sets of size exactly 2.

For each homologous set $R = R_1, \ldots, R_k$ with $|R| > 2$ we do the following: Let $R = \{(v_0, v_1), (v_0, v_2), \ldots, (v_0, v_p)\}$. Each arc in R emanates from node v_0. In the digraph D, these arcs are replaced with the structure illustrated in Figure 2.2. Formally, we introduce p new nodes $v'_1, \ldots, v'_p$. Then, we replace the arc (v_0, v_i) with two arcs $(v_0, v'_i), (v'_i, v_i)$ for all $i = 1, \ldots, p$. All these arcs are assigned a capacity of 1. The set R itself is replaced by the sets $\{(v'_i, v_i), (v_0, v'_{i+1})\}$ for all $i = 1, \ldots, p-1$ and the set $\{(v'_p, v_p), (v_0, v'_1)\}$.

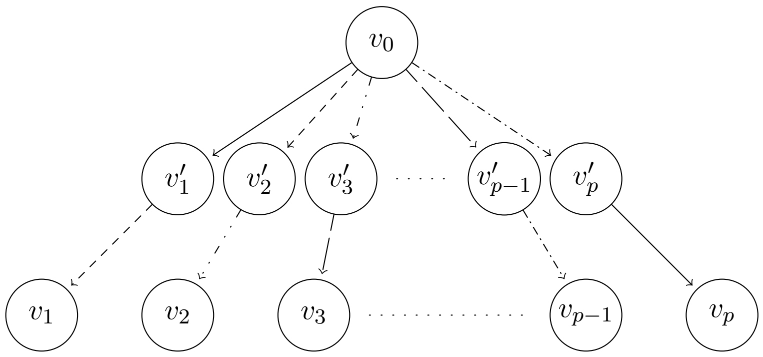


Figure 2.2.: a) Structure by which a homologous set $\{(v_0, v_1), \ldots, (v_0, v_p)\}$ is transformed into p homologous sets of size 2. Two edges with the same pattern belong to the same homologous set.

Together with the flow preservation constraints the new homologous sets ensure that all arcs $(v_0, v'_i), (v'_i, v_i)$, for $i = 1, \ldots, p$, carry the same amount of flow. As all arcs in R are replaced by simple paths of length 2, the set of feasible flows in D is not affected. Hence, each feasible flow in the original problem instance can be easily transformed into a feasible flow of the modified problem instance with the same value, and vice versa.

The total number of arcs in all sets $R_1, \ldots, R_k$ is bounded above by the total number of arcs in D. Therefore, the number of additional nodes added to D is at most $|A|$, the total number of new arcs is at most $2|A|$ and the number of homologous sets of size 2 is also bounded by $|A|$. This shows that the reduction described here is polynomial in the size of the digraph D. $\square$

Transformation of the CCMP on bipartite graphs and the LCMP into the maximum paired integer equal flow problem

The condition that all arcs in a homologous set must carry the same amount of flow allows us to transform the CCMP on bipartite graphs and the LCMP into the maximum paired integer equal flow problem. Both transformations are based on the standard technique of transforming a bipartite matching problem into a maximum flow problem, which is described in Section 1.3.1. We begin with the transformation of the CCMP on bipartite graphs, as the transformation of the LCMP is based on it.

Let $(G, \mathcal{F})$ be a problem instance of the CCMP, with $G = (U \cup V, E)$ being a bipartite graph and $\mathcal{F} = \{F_1, \ldots, F_p\}$ being a couple collection on the set of edges E. The maximum paired integer equal flow problem to solve this CCMP instance is posed on the digraph $D = (U \cup V \cup \{s, t\}, A)$, with a source node s and a sink node t. The set of arcs is defined as follows:

$$A := \{(s,u) \mid u \in U\} \cup \{(v,t) \mid v \in V\} \cup \{(u,v) \mid [u,v] \in E \text{ with } u \in U, v \in V\}\,.$$

The capacity of all arcs in A is set to 1. The homologous sets $R_1, \dots, R_p$ for this problem are defined such that the corresponding homologous constraints reflect the couple constraints:

$$R_i := \{(u,v) \mid [u,v] \in F_i \text{ with } u \in U, v \in V\} \quad \text{for all } i = 1, \dots, p.$$

All sets $R_1, \dots, R_k$ are of size 2. Hence, $(D, s, t, R_1, \dots, R_p)$ is an instance of the maximum paired integer equal flow problem.

Analogous to the standard technique of transforming a bipartite matching problem into a maximum flow problem there is an equivalence between feasible matchings of the CCMP and flows of the paired integer equal flow problem: There is a matching of size v in G satisfying the couple constraints concerning $\mathcal{F}$ if and only if there is an (s,t)-flow of value v in D with all arcs in a set R_i carrying the same amount of flow for all $i = 1, \dots, p$.

Let us now consider the LCMP. Instances of this problem type can also be transformed into maximum paired integer equal flow problem instances with all arcs having a capacity of 1. The transformation mainly builds on the reformulation of the level constraint as a set of couple constraints. We refer to Section 5.1 in which we show how to transform an LCMP instance into a CCMP instance. Then, the resulting CCMP instance can be transformed into an instance of the maximum paired integer equal flow problem as described above.

Regarding the complexity of the integer equal flow problem, it is important whether the number of homologous sets is fixed or not (see Section 2.3 for discussions about the impact of fixed problem parameters to combinatorial optimization problems). The maximum integer equal flow problem and the more general minimum cost flow version of it are polynomially solvable if the number of homologous sets is fixed (see [41]). Ahuja et al. [5] present explicit solution algorithms for the special case where the minimum cost integer equal flow problem contains exactly one homologous set.

2.4.2. Couple constraints and the symmetric matching problem

Definition 2.24. *Let $G = (U \cup V, E)$ be a bipartite graph with $U = \{u_1, \dots, u_n\}$ and $V = \{v_1, \dots, v_n\}$. Furthermore, let the index of each node in U and V be fixed. The graph G is called* symmetric *if $[u_i, v_j] \in E \Leftrightarrow [u_j, v_i] \in E$ for all $i, j = 1, \dots, n$.*

We now consider a special class of CCMP instances, where the underlying graph $G = (U \cup V, E)$ is a symmetric bipartite graph with $U = \{u_1, \dots, u_n\}$, $V = \{v_1, \dots, v_n\}$. Further, the couples in the couple collection $\mathcal{F} = \{F_1, \dots, F_p\}$ are such that a solution matching in G is symmetric, i.e. for each symmetric pair of edges $([u_i, v_j], [u_j, v_i]) \in E \times E$ there is a couple $\{[u_i, v_j], [u_j, v_i]\} \in F$ and each couple in $\mathcal{F}$ corresponds to a symmetric pair of edges in $E \times E$. In this case, the CCMP is equivalent to the *symmetric matching problem* (see [37]). This problem can be solved by transforming it into a maximum matching problem. One should note that in our case the graph G may contain edges of the form $[u_i, v_i]$, in contrast to the case discussed by Lawler [37].

In the proof of Theorem 2.26 it is shown how the transformation works for the weighted case, i.e. for finding a maximum weight symmetric matching in a symmetric graph G. We formulate this problem in the more general context of weighted couple constrained matching problems. Of course, this method can also be used for the unweighted case.

Let $c_e \in \mathbb{R}$ be the weight of the edge e for all $e \in E$.

Problem Formulation 2.25 (Weighted couple constrained matching)**.** *The* weighted couple constrained matching problem (w-CCMP) *is stated as follows:*

$$\begin{aligned} \max \quad & \sum_{e\in E} c_e x_e \\ s.t. \quad & (2.2) - (2.4). \end{aligned}$$

Theorem 2.26. *Let $G = (U \cup V, E)$ be a symmetric bipartite graph. Let $\mathcal{F}$ be a couple collection in G with $F = \{e, f\} \in \mathcal{F}$ if and only if e and f are symmetric edges in E. Furthermore, let $c_e \in \mathbb{R}$ be the weight of the edge e for all $e \in E$.*

Then, the weighted couple constrained matching problem with couple collection $\mathcal{F}$ can be solved in polynomial time.

Proof. Let $(G, \mathcal{F}, c)$ be an instance of the w-CCMP, where $G = (U \cup V, E)$ is a symmetric bipartite graph, $\mathcal{F}$ is a couple collection with the property that $F = \{e, f\} \in \mathcal{F}$ if and only if e and f are symmetric edges in E, and $c \in \mathbb{R}^E$ is a vector of edge weights.

The transformation of $(G, \mathcal{F}, c)$ into a problem instance $(H = (W, M), d)$ of the general weighted matching problem works as follows. Each pair of nodes $(u_i, v_i) \in U \times V$ induces a node w_i in W. Further, each pair of symmetric edges $([u_i, v_j], [u_j, v_i]) \in E \times E$ with $i \neq j$ induces an edge $[w_i, w_j]$ in M, which is assigned the weight $d_{[w_i,w_j]} = c_{[u_i,v_j]} + c_{[u_j,v_i]}$. An edge of the form $[w_i, w_j]$ in H represents the two symmetric edges $[u_i, v_j]$ and $[u_j, v_i]$ in G.

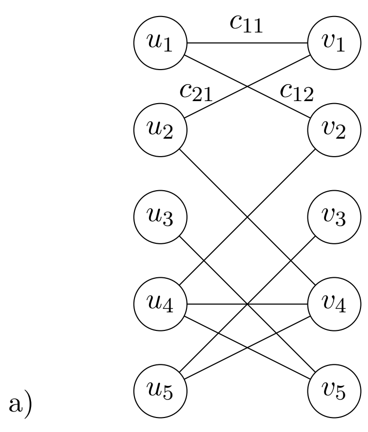
For each edge $[u_i, v_i]$ in E, the graph H further contains a node x_i and an edge $[w_i, x_i]$ with weight $d_{[w_i,x_i]} = c_{[u_i,v_i]}$. The edge $[w_i, x_i]$ represents the single edge $[u_i, v_i]$ in G. Figure 2.3 depicts an example of the construction of the graph H.

Now, G contains a matching M satisfying the couple constraints concerning $\mathcal{F}$ which is of total weight z if and only if H contains a matching N of total weight z. $\square$

It is worth mentioning that if there are no edges of the form $[u_i, v_i]$ in G, then the graph H in the proof of Theorem 2.26 does not contain any nodes x_i. Hence, if we have an unweighted CCMP with a couple collection with properties as in Theorem 2.26, stated on a symmetric bipartite graph G which does not contain any edges of the form $[u_i, v_i]$, then the transformation results in an unweighted matching problem.

2.4.3. The level constraint and the symmetric matching problem

Let us consider the special case in which, besides satisfying the level constraint, the sought-after perfect matching in the LCPMP must also be symmetric. The underlying graph G for this kind of problem is a symmetric bipartite graph.



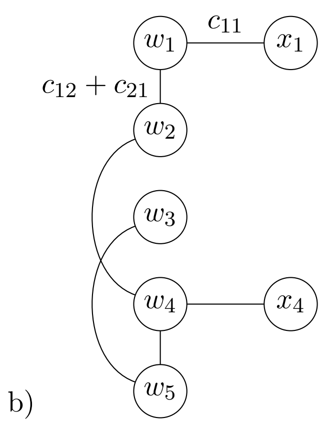


Figure 2.3.: a) Example of a weighted symmetric bipartite graph. c_{ij} denotes the weight of edge $[u_i, v_j]$ (weights only partially listed). b) Weighted graph resulting from the problem transformation. Weights of edges are noted along the edges (weights only partially listed).

Problem Formulation 2.27 (Level constrained symmetric perfect matching)**.** *The* level constrained symmetric perfect matching problem (LCPMP-SYM) *is the problem of finding a symmetric perfect matching satisfying the level constrained in a symmetric bipartite graph.*

Theorem 2.28. *The level constrained symmetric perfect matching problem can be solved in polynomial time.*

Proof. Let (G, k) be an instance of the LCPMP-SYM, where $G = (U \uplus V, E)$ is a symmetric bipartite graph with $U = \{u_1, \dots, u_n\}, V = \{v_1, \dots, v_n\}$, and k is a nonnegative integer with $k \leq n$. This problem instance transforms to a perfect matching problem on a (not necessarily symmetric) graph $H = (W, F)$, which is defined as follows (see Figure 2.4):

$$W := \{w_1, \dots, w_n, x_1, \dots, x_k\}, \tag{2.20}$$

$$\begin{aligned} F := \{[w_i, w_j] \mid ([u_i, v_j], [u_j, v_i]) \text{ is a pair of symmetric edges in } G \text{ with } i \neq j\} \\ \cup \{[w_i, x_j] \mid [u_i, v_i] \in E, j = 1, \dots, k\}. \end{aligned} \tag{2.21}$$

Each pair of symmetric edges $([u_i, v_j], [u_j, v_i])$, with $i \neq j$, induces an edge $[w_i, w_j]$ in the graph H. Hence, each edge of the form $[w_i, w_j]$ in H represents the two symmetric edges $[u_i, v_j]$ and $[u_j, v_i]$ in G. For each on-level edge $[u_i, v_i]$ in G, the node w_i is connected to all nodes $x_1, \dots, x_k$ in the graph H. Each edge $[w_i, x_j], j = 1, \dots, k$, stands for the possibility to choose the on-level edge $[u_i, v_i]$ to be in a symmetric matching in G.

Now, we show that there is a symmetric perfect matching M in G which contains exactly k on-level edges if and only if there is a perfect matching N in H. Let M be a symmetric perfect matching in G containing exactly k on-level edges. Without loss of generality, let $[u_1, v_1], \dots, [u_k, v_k]$ be those on-level edges (otherwise, the nodes in G can

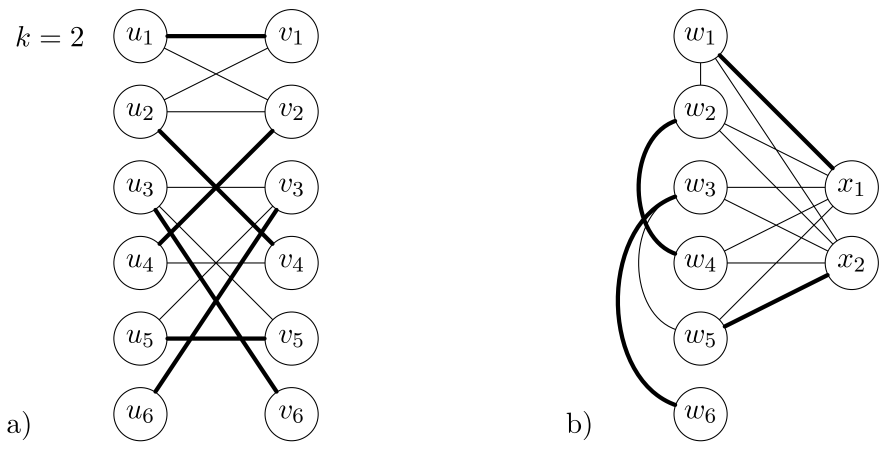


Figure 2.4.: a) Example of a problem instance of the LCPMP-SYM with level constraint parameter $k = 2$. A solution matching M is indicated with bold edges. b) Graph resulting from problem reduction. The bold edges indicate a perfect matching N.

be renamed taking into account that $u_i \in U$ is renamed $u_j \in U$ if and only if $v_i \in V$ is renamed $v_j \in V$). Based on M, the set N is defined as

$$N := \{[w_i, w_j] \mid ([u_i, v_j], [u_j, v_i]) \text{ is a symmetric pair of edges in } M \text{ with } i \neq j\} \cup \{[w_i, x_i] \mid [u_i, v_i] \in M\}. \tag{2.22}$$

It holds that N is a matching in H, as by definition, no two edges of N meet at a node x_i. Further, no two edges share an end-node w_i, as no two edges in M share an end-node u_i or v_i. What remains to be shown is that N covers all nodes in H. The matching M is perfect and hence covers the nodes u_i and v_i for all $i = 1, \ldots, n$. Therefore, N covers all nodes $w_i, i = 1, \ldots, n$. Furthermore, the node x_j is covered by N for all $j = 1, \ldots, k$, which results from the definition of N and the fact that M contains the k on-level edges $[u_1, v_1], \ldots, [u_k, v_k]$.

Now, let N be a perfect matching in H. We construct a matching M in the graph G conversely to the construction of N from M.

$$M := \{[u_i, v_j], [u_j, v_i] \mid [w_i, w_j] \in N\} \cup \{[u_i, v_i] \mid \exists\, j : [w_i, x_j] \in N\}. \tag{2.23}$$

By definition, M is symmetric. As no two edges in N meet at a node w_i, no two edges in M meet at a node u_i or v_i. Hence, the set M is a matching. Each N-covered node w_i implies that the nodes u_i and v_i both are M-covered. Since all nodes w_i for $i = 1, \ldots, n$ are N-covered, it follows that M is a perfect matching. What remains to be shown is that M contains exactly k on-level edges. This holds true, as N is perfect and hence covers all nodes x_j, for $j = 1, \ldots, k$, by k edges of the form $[w_i, x_j]$.

The fact that the perfect matching problem can be solved in polynomial time (see Section 1.3) finishes the proof. □

The statement in Theorem 2.28 can be extended to the weighted case, where a symmetric matching satisfying the level constraint and being of maximum weight is sought.

Problem Formulation 2.29 (Weighted level constrained symmetric matching)**.** *The* weighted level constrained symmetric matching problem (w-LCMP-SYM) *is the problem of finding a maximum weight symmetric matching satisfying the level constraint in a symmetric graph.*

Theorem 2.30. *The weighted level constrained symmetric matching problem is polynomially solvable.*

Proof. We transform the w-LCMP-SYM into a general weighted matching problem. For that, let (G, c, k) be an arbitrary instance of the w-LCMP-SYM, where $G = (U \,\dot\cup\, V, E)$ is a symmetric bipartite graph with $U = \{u_1, \ldots, u_n\}$, $V = \{v_1, \ldots, v_n\}$, $c : E \to \mathbb{R}$ is a weight function and k is a nonnegative integer with $k \leq n$. This problem instance is transformed into a problem instance (H, c') of the weighted matching problem. The graph $H = (W, F)$ is constructed as described by (2.20) and (2.21) in the proof of Theorem 2.28. Hence, its node set W contains the nodes w_i for all $i = 1, \ldots, n$ and the nodes x_j for all $j = 1, \ldots, k$.

The edge weights c' are defined based on the weight function c of the graph G. An edge of the form $[w_i, w_j] \in F$ is assigned the weight

$$c'_{[w_i,w_j]} := c_{[u_i,v_j]} + c_{[u_j,v_i]} \quad \text{for all } i, j = 1, \ldots, n \text{ with } i \neq j.$$

In order to define the weights of the remaining edges in H, let $M := \sum_{e \in E} |c_e| + 1$. Then, an edge of the form $[w_i, x_j] \in F$ is assigned the weight

$$c'_{[w_i,x_j]} := c_{[u_i,v_i]} + M,$$

for all $i = 1, \ldots, n$ for which the on-level edge $[u_i, v_i]$ exists in G.

We now show that the graph G has a symmetric matching of weight z containing exactly k on-level edges if and only if the graph H has a matching of weight $z + kM$. In order to transform a given symmetric matching N_G in G into a matching N_H in H, and vice versa, we will use the same construction method as described in (2.22) and (2.23) in the proof of Theorem 2.28.

The transformation of N_G into N_H and vice versa is based on the assumption that the k on-level edges in N_G are the edges $[u_1, v_1], \ldots, [u_k, v_k]$. This can be assumed without loss of generality (otherwise, the nodes are renamed by taking into account that $u_i \in U$ is renamed $u_j \in U$ if and only if $v_i \in V$ is renamed $v_j \in V$). We have seen in the proof of Theorem 2.28 that N_G is a symmetric matching if and only if N_H is a matching.

Concerning the weights of the matchings we have the following situation: For an edge f in H with an end-node x_i, its weight c'_f is the weight of the corresponding on-level edge in G plus the summand M. As the symmetric matching N_G contains exactly k on-level edges, the matching N_H contains exactly k edges which have an end-node of the form x_i. The weights of the other edges in N_H simply accumulate the weights of the corresponding symmetric edges in N_G. Hence, if N_G has weight z, then N_H has weight $z + kM$.

To show the other direction of the equivalence, let N_H be a matching in H which has weight $z + kM$. The weight of N_H and the definition of M imply that N_H contains exactly k of the edges with end-nodes $x_1, \dots, x_k$. We construct the symmetric matching N_G from N_H following the procedure described in (2.23) in the proof of Theorem 2.28. Thus, N_G contains exactly k on-level edges. It further holds that all edges in N_H of the form $[w_i, w_j]$ induce two edges in N_G whose weights sum up to $c'_{[w_i,w_j]}$ and all edges in N_H of the form $[w_i, x_j]$ induce an edge in N_G of weight $c'_{[w_i,x_j]} - M$. Therefore, if N_H has weight $z + kM$ then N_G has weight z. □

Equivalence between symmetric perfect matchings with satisfied level constraint and general perfect matchings

We use the transformation from the proof of Theorem 2.28 to establish a one-to-one correspondence between symmetric perfect matchings with satisfied level constraint and general perfect matchings. For that, we need to exclude the possibility that one symmetric perfect matching can be represented by more than one general perfect matching. This situation occurs in the transformation presented in the proof of Theorem 2.28.

To see this, let $G = (U \mathbin{\dot\cup} V, E)$ be a symmetric bipartite graph with $U = \{u_1, \dots, u_n\}$, $V = \{v_1, \dots, v_n\}$ and let k be a nonnegative integer with $k \leq n$. Let the graph $H = (W, F)$ be constructed as described by (2.20) and (2.21) in the proof of Theorem 2.28. We consider a perfect matching N in the graph H. Let

$$L(N) := N \cap \{[w_i, x_j] \mid i = 1, \dots, n; j = 1, \dots, k\}$$

be the set of those edges in N which have an end-node in $\{x_1, \dots, x_k\}$.

Let $\pi : \{1, \dots, k\} \to \{1, \dots, k\}$ be an arbitrary permutation of the indices $1, \dots, k$. Permuting the end-nodes x_i of the edges in $L(N)$ according to π yields

$$L^\pi(N) := \left\{[w_i, x_{\pi(j)}] \mid [w_i, x_j] \in L(N)\right\}.$$

Based on the permutation π, let N^π be the matching resulting from permuting the end-nodes x_i of the edges in $L(N)$:

$$N^\pi = N \setminus L(N) \cup L^\pi(N).$$

One should note that all nodes $w_i, i = 1, \dots, n$, are covered by N^π. Further, all nodes $x_i, i = 1, \dots, k$, are N^π-covered. Hence, the set N^π is a perfect matching in H. Now, according to the transformation in (2.23), the matching N^π transforms to the same symmetric perfect matching in G for all permutations π.

In order to avoid that a symmetric perfect matching in G is represented by more than one perfect matching in H, we consider a restriction on the perfect matchings in H: Let N be a perfect matching in H and let

$$I(N) := \{i \mid \exists\, j \in \{1, \dots, k\} : [w_i, x_j] \in N\}$$

be the set of those indices i for which the nodes w_i are covered by edges incident to any nodes $x_j, j = 1, \ldots, k$. Let $\phi_N : I(N) \to \{1, \ldots, k\}$ be defined by $\phi_N(i) := j$, where j is such that $[w_i, x_j] \in L(N)$. The mapping ϕ_N is well-defined, as for each $i \in I(N)$ there is exactly one edge of the form $[w_i, x_j]$ in the perfect matching N. Now, we allow only those perfect matchings N in H for which ϕ_N is strictly increasing, i.e. $\phi(i) < \phi(j)$ for all $i, j \in I$ with $i < j$.

Lemma 2.31. *Let $G = (U \cup V, E)$ be a symmetric graph with $U = \{u_1, \ldots, u_n\}$, $V = \{v_1, \ldots, v_n\}$ and let k be a nonnegative integer with $k \leq n$. Let $H = (W, F)$ be the graph defined as in (2.20) and (2.21). Further, let $\mathcal{M}$ be the set of symmetric perfect matchings in G which contain exactly k on-level edges and let $\mathcal{N}$ be the set of perfect matchings N in H for which ϕ_N is strictly increasing. Then, there exists a bijection $\tau : \mathcal{M} \to \mathcal{N}$.*

Proof. First, we specify the mapping τ. For that, we need to ensure that all matchings $N = \tau(M)$ satisfy that ϕ_N is strictly increasing. Let M be a matching in $\mathcal{M}$. Let $[u_{i_1}, v_{i_1}], \ldots, [u_{i_k, v_{i_k}}]$ be the on-level edges in M, ordered by strictly increasing indices of their end-nodes, i.e. $i_h < i_l$ for all $h < l$.

Then, τ is defined by $\tau(M) := N$ with

$$\begin{aligned} N := \{[w_i, w_j] \mid ([u_i, v_j], [u_j, v_i]) \text{ is a symmetric pair of edges in } M, i \neq j\} \\ \cup \{[w_{i_j}, x_j] \mid j = 1, \ldots, k\}. \end{aligned}$$

We have already seen that N is a perfect matching in H. In the definition of N we make use of the fact that we have ordered the on-level edges of M with strictly increasing indices of their end-nodes. As this order is unique, the mapping τ is well-defined. The strictly increasing order of the on-level edges in M implies that the nodes $w_{i_1}, \ldots, w_{i_k}$ are also ordered with strictly increasing indices. Therefore, it holds that ϕ_N is strictly increasing and hence, $N \in \mathcal{N}$. Figure 2.4 shows an example of a transformation of a matching $M \in \mathcal{M}$ into a matching $\tau(M) \in \mathcal{N}$.

To show the injectivity of τ, let $M_1, M_2 \in \mathcal{M}$ be such that $\tau(M_1) = \tau(M_2) = N$. All off-level edges in M_1 and M_2 appear only in symmetric pairs, and each symmetric pair of off-level edges uniquely induces one edge in N. Therefore, both matchings M_1 and M_2 must contain the same off-level edges. The set of edges $[w_{i_j}, x_j], j = 1, \ldots, k$ in N is induced by a unique set of on-level edges. Hence, M_1 and M_2 contain the same on-level edges. So, it holds that $M_1 = M_2$.

To show the surjectivity of τ, let N be an arbitrary matching in $\mathcal{N}$. We define a matching $M \in \mathcal{M}$ as follows:

$$M := \{[u_i, v_j], [u_j, v_i] \mid [w_i, w_j] \in N\} \cup \{[u_i, v_i] \mid \exists\, j \in \{1, \ldots, k\} : [w_i, x_j] \in N\}.$$

Obviously, $\tau(M)$ contains an edge $[w_i, w_j]$ if and only if $[w_i, w_j] \in N$. Further, M contains an on-level edge $[u_i, v_i]$ if and only if node w_i is N-covered. Therefore, $\tau(M)$ and N cover the same nodes w_i by an edge incident to nodes x_i. Since the nondecreasing order of these nodes by their indices is unique, both matchings $\tau(M)$ and N contain the same edges incident to nodes x_i. Finally, the matchings $\tau(M)$ and N contain the same edges and therefore, $\tau(M) = N$. □

2.4.4. Level constrained perfect matching and 3-dimensional matching

As stated in Sections 1.3 and 1.5, the perfect matching problem is a polynomially solvable problem and the 3-dimensional matching problem is NP-hard. It is an interesting question of what complexity problems situated "between" these two problem types are, i.e. problems which cannot be modeled as perfect matching problems but which are special cases of 3DM. The level constrained perfect matching problem is such a problem. The LCPMP can be interpreted as a 3DM problem where the matching constraints concerning the nodes in its third color class are used only to ensure that the level constraint is satisfied.

We now show that the LCPMP is a special case of 3DM.

Theorem 2.32. *The level constrained perfect matching problem can be polynomially reduced to the 3-dimensional matching problem.*

Proof. Let $(G = (U \cup V, E), k)$ be an instance of the LCPMP, where G is a bipartite graph with $U = \{u_1, \dots, u_n\}, V = \{v_1, \dots, v_n\}$ and k is an integer-valued number with $k \leq n$. This instance is reduced to a 3DM problem instance on the hypergraph $H = (U \cup V \cup W, \mathcal{E})$, where $W := \{w_1, \dots, w_n\}$. Its set of edges is defined by

$$\begin{aligned}\mathcal{E} :=& \Big\{\{u_i, v_i, w_h\} \mid [u_i, v_i] \in E_{\text{on}} \text{ with } u_i \in U, v_i \in V; h = 1, \dots, k\Big\}\\ &\cup \Big\{\{u_i, v_j, w_h\} \mid [u_i, v_j] \in E_{\text{off}} \text{ with } u_i \in U, v_j \in V; h = k+1, \dots, n\Big\},\end{aligned}$$

where E_{on} denotes the set of on-level edges and E_{off} denotes the set of off-level edges in the graph G.

Now, let M be a perfect matching in G containing exactly k on-level edges. Without loss of generality, let $[u_1, v_1], \dots, [u_k, v_k]$ be the on-level edges in M. Let $\mathcal{M} \subseteq \mathcal{E}$ be defined as

$$\begin{aligned}\mathcal{M} :=& \Big\{\{u_i, v_i, w_i\} \mid i = 1, \dots, k\Big\}\\ &\cup \Big\{\{u_i, v_j, w_i\} \mid [u_i, v_j] \in M \text{ with } u_i \in U, v_j \in V \text{ and } i \neq j\Big\}.\end{aligned}$$

As M is a matching and all nodes w_i appearing in any edge of $\mathcal{M}$ are pairwise different, there are no edges in $\mathcal{M}$ sharing a node. Thus, $\mathcal{M}$ is a matching in the hypergraph H. As M contains exactly n edges, $\mathcal{M}$ contains n edges as well. Hence, it is a solution to the 3DM problem on H.

Next, let $\mathcal{M}$ be a solution to the 3DM problem on the hypergraph H. We define a set $M \subseteq E$ as follows:

$$\begin{aligned}M :=& \Big\{[u_i, v_i] \mid u_i \in U, v_i \in V : \{u_i, v_i, w_h\} \in \mathcal{M} \text{ for some } h = 1, \dots, k\Big\}\\ &\cup \Big\{[u_i, v_j] \mid u_i \in U, v_j \in V : \{u_i, v_j, w_h\} \in \mathcal{M} \text{ for some } h = k+1, \dots, n\Big\}\end{aligned}$$

For each edge in M both of its end-nodes also appear in an edge in the matching $\mathcal{M}$. Further, no two edges in $\mathcal{M}$ share a node. Thus, the set M is a matching in G. The hypergraph H is constructed such that each edge in $\mathcal{E}$ contains a node in W. Together with the fact that $\mathcal{M}$ is a matching in H of size n, this implies that M is of size n as well. Further, all nodes in W are covered by $\mathcal{M}$. Each edge in $\mathcal{M}$ that covers a node w_h for any $h = 1, \dots, k$ induces an on-level edge $[u_i, v_i]$ for some $i = 1, \dots, n$ in M. Each edge in $\mathcal{M}$ that covers a node w_h for any $h = k+1, \dots, n$ induces an off-level edge in M. Thus, M contains exactly k on-level edges.

The construction of the hypergraph H and the construction of the matching M based on $\mathcal{M}$ can be done polynomially in the size of G. Hence, the reduction is polynomial, which completes the proof. □

2.4.5. Matching problems with different variations of additional side constraints in the literature

There are numerous variations of matching problems with additional side constraints considered in the literature. Our aim here is to present those whose additional side constraints are closely related to couple constraints and the level constraint, rather than giving a complete list of matching problems with additional side constraints. For overviews of these problems, the reader may refer to e.g. [47] and [11].

A natural generalization of the couple constrained matching problem arises from allowing couples to be of arbitrary size. This case is considered by Padberg and Sassano in [45], where the generalized couples are called bonds. The authors introduce the problem *matching with bonds*, where bonds replace couples in the corresponding constraints. Matching with bonds is further investigated in Section 3.1.

A formulation of additional side constraints by means of a digraph is given by Hefner and Kleinschmidt [28]. They introduce the *Master/Slave-matching problem*, whose problem input consists of a graph $G = (V, E)$ and a digraph $D = (V, A)$ on the same set of nodes. The aim is to find a maximum matching M in G, such that for all arcs $(u, v) \in A$ the following holds: If u is M-covered then v is M-covered. The authors show that the Master/Slave-matching problem is NP-hard.

Concerning the level constraint there are more problems with variations of this constraint in the literature. The level constraint is an equality constraint referring to a special subset of edges of a bipartite graph. A natural generalization is to demand this equality constraint for a general subset of edges. Imposing this type of constraint on a perfect matching problem yields the *equality constrained perfect matching problem* (ECPMP), which is defined as follows. Let $G = (U \cup V, E)$ be a bipartite graph with $|U| = |V| = n$, let $R \subseteq E$ be a general subset of the edges in G and let k be an integer-valued parameter with $0 \leq k \leq n$. The task is to find a perfect matching M in G satisfying the additional constraint

$$|M \cap R| = k. \tag{2.24}$$

Here, R is part of the problem input and hence it is not fixed. The problem was formulated by Papadimitriou and Yannakakis in [46]; its complexity is still unknown (refer to [7], [52] and [16]). We review the results of Papadimitriou and Yannakakis [46] and establish a complexity relationship to the LCPMP in Section 4.1.

It is interesting to see that the ECPMP is a nontrivial problem even on complete bipartite graphs. This problem can be interpreted as finding an assignment which satisfies the additional side constraint (2.24); it is called the *equality constrained assignment problem* (ECAP). Similar to the last problem, the task here is to find a perfect matching M in a complete bipartite graph $K_{n,n}$ satisfying the side constraint $|M \cap R| = k$, where R is a subset of the edge set of the graph $K_{n,n}$. We further discuss this problem in Section 4.3. A polynomial time algorithm for solving this problem is given by Karzanov [32] and Yi et al. [52].

If the additional side constraint in the ECPMP is replaced by an inequality of the same structure, it is not required that an exact number of edges in a perfect matching are elements of R, but rather there is a defined upper bound on the number of edges which are allowed to be elements of R. The *restricted perfect matching problem* (RPMP) is the problem of finding a perfect matching M in a bipartite graph $G = (U \uplus V, E)$ satisfying the upper bound constraints

$$|M \cap R_i| \leq r_i,$$

with $R_i \subseteq E$ and r_i being integral for all $i = 1, \dots, l$. This problem is investigated in [29] and [46]. We discuss its complexity relationship to the LCPMP in Section 4.1. The consequences of a fixed number of upper bound constraints as well as of fixed bounds on the values r_i are discussed in Section 4.2.

Concerning the ECPMP, Costa et. al [16] study this problem with a focus on the characterization of the set R, for which a matching satisfying the additional constraint exists. More specifically, given a sequence of integers $k_1, \dots, k_s$, they search for a smallest subset of edges R such that there exists a maximum matching M_i in G satisfying $|M_i \cap R| = k_i$ for all $i = 1, \dots, s$. The authors prove the minimum cardinality of R in the case of regular bipartite graphs with two parameters k_1 and k_2. Beside giving results for special cases, they also present upper bounds on the cardinality of R for the case of $k_1, \dots, k_s$ being consecutive integers.

Side constraints which directly affect the notion of a matching are investigated by Stockmeyer and Vazirani [51]. For a graph G and an integer-valued parameter δ they ask for a matching M in G which is of maximum cardinality and which has the property that the distance between each pair of edges in M is at least δ. The authors call this problem *maximum δ-separated matching.* They show that the decision version of this problem is NP-complete, even for bipartite graphs with a maximum node degree of 4.

Most of the matching-type problems with additional side constraints discussed in the literature are assignment problems.

Dell'Amico and Martello [18] introduce the *k-cardinality assignment problem* (k-AP). In this problem, all feasible assignments are restricted to be of size k. Hence, the task is to find a minimum cost assignment among all those assignments that are of cardinality k. The cardinality restriction is ensured by one additional side constraint. This side constraint is an equality constraint referring to all edges in a complete bipartite graph. The main results to this problem are discussed in Section 4.3. As applications of this problem the authors suggest the problem of finding a minimum cost assignment of workers to machines, under the condition that only a subset of workers and machines needs to be assigned. Further, they mention that the k-AP may appear as a subproblem when solving a specific satellite communication problem. In this problem, m earth stations need to communicate to n different earth stations via satellite. The satellite is using a $k \times k$ switch, which allows to establishes a specific set of k interconnections at a time.

A well known generalization of the classical assignment problem is the *generalized assignment problem* (GAP). Here, the assignment problem has several upper bound side constraints imposed. The GAP can be interpreted as the problem of assigning jobs $j \in J = \{1, \dots, m\}$ to agents $i \in I = \{1, \dots, n\}$, with the restrictions that each job is assigned to exactly one agent, capacity restrictions on the agents are fulfilled and the total cost of the assignment is minimized. The problem is formulated as follows:

$$\begin{aligned} \min \quad & \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij} & \\ \text{s.t.} \quad & \sum_{i \in I} x_{ij} = 1 & \forall\, j \in J \\ & \sum_{j \in J} a_{ij} x_{ij} \le b_i & \forall\, i \in I \\ & x_{ij} \in \{0,1\} & \forall\, i \in I, j \in J, \end{aligned} \tag{2.25}$$

where $c_{ij} \in \mathbb{R}$ is the cost of assigning job j to agent i, the value $a_{ij} \in \mathbb{R}$ is the capacity consumption of job j being processed by agent i, and $b_i \in \mathbb{R}$ is the capacity limit of agent i, for all $i \in I$ and $j \in J$. The additional restrictions (2.25) are knapsack constraints. Their number is not fixed and it can grow linearly with the number of agents. One can see directly that the assignment problem in its original form (see Section 1.2.3) results from the GAP when $|I| = |J|$ and $a_{ij} = 1, b_i = 1$ for all $i \in I, j \in J$. Fisher et. al [24] prove that the GAP is an NP-hard problem. In [12], Cattrysse and Wassenhove give an overview of different kinds of relaxations of the GAP and compare the bounds these relaxations deliver.

A survey of real-life applications where the GAP appears is given by Öncan [44]. These applications comprise employee and machine scheduling, routing and supply chain problems, as well as different variants of the facility location problem. Further, the author presents several extensions of the GAP, e.g. the non-linear capacity constrained GAP or a multi-resource GAP which has multiple knapsack constraints per agent.

Mazzola and Neebe [39] discuss the *assignment problem with side constraints* (APSC), which generalizes the GAP. Additionally to the constraints used in the classical assignment problem, the APSC has the constraints

$$\sum_{i \in I} \sum_{j \in J} a_{ijk} x_{ij} \le b_k \quad \text{for all } k \in S, \tag{2.26}$$

with $S = \{1, \dots, s\}$. The authors present a branch-and-bound procedure for solving the APSC to optimality. For that, lower bounds on the optimal objective function value are obtained by solving the Lagrangian relaxation of the APSC, where the constraints (2.26) are dualized. Further, the authors develop a heuristic for approximating an optimal solution to the APSC. Aboudi and Jörnsten investigate the same problem in [2]. They call it the *resource constrained minimum weight assignment problem.*

Beside problem variants where the task is to find an assignment of minimum cost, a common problem variant asks for an assignment of specific cost. Clearly, this generalizes a side constraint which demands a specific number of edges to be out of a given set $R \subseteq E$. Kochenberger et. al [33] present an approach where the additional side constraint demanding the exact weight property is transformed into a quadratic penalty objective function. The resulting problem is the *quadratic assignment problem* (QAP), for which the authors present computational results based on a heuristic.

A different approach to include additional restrictions in an assignment problem is followed by Felici and Mecoli [23]. They introduce the so-called *assignment problem with preference conditions* (APPC). The preference conditions only affect the objective function value of an assignment and have no impact on its feasibility. Hence, in the APPC an assignment which satisfies the preference conditions is of lower cost than one that does not satisfy them. The authors consider two types of preference constraints: The *join preference*, which demands that all edges in a predefined subset of edges are in the assignment, and the *split preference*, which demands that at most one edge of a predefined subset of edges is in the assignment. In practical applications, the preference constraints may arise in the following way: In jobs to workers assignments, the constraints can model the situation where it is beneficial that groups of workers are working together on jobs or that, on the contrary, no two workers of the same group are working on a job. In the context of job shop problems, the preference constraints allow us to consider situations where all jobs of a specific type should be processed on the same machine, or situations where conflicts arise when certain jobs are processed on the same machine. In an integer programming formulation of the APPC in [23], the impact of preference constraints on the objective function is modeled by means of additional binary variables. The authors show that the APPC with split preferences can be solved in polynomial time by transformation into a minimum cost flow problem. The APPC with join preferences is NP-hard. For this problem variant, a heuristic which is also based on a minimum cost flow problem is presented.

There are further restrictions which only affect the objective function. In the case that the objective is not to minimize the total cost of an assignment but rather to minimize the maximum cost of one edge appearing in an assignment, we get the *bottleneck assignment problem.* Its objective function is

$$\min \max_{i,j=1,\dots,n} c_{ij} x_{ij}.$$

The bottleneck assignment problem is investigated by Mazzola and Neebe in [40]. Of course, there are several different variants of similar objective functions, e.g. for minimizing the maximum difference between the costs of two edges in an assignment or for minimizing the sum of the k largest costs of edges appearing in an assignment (see [47] for an overview).

In [25], Gardner et. al discuss various problems in discrete tomography. They cover complexity questions regarding the existence, uniqueness and the reconstruction of finite subsets of the lattice $\mathbb{Z}^d$ from their line sums in m lattice directions. A lattice point is in the subset if it has assigned a value of 1, otherwise its value is 0. A line sum is the sum of values of the lattice points which are on a given line. For each given direction $v \in \mathbb{Z}^d \setminus \{0\}$ a line sum is given for all lines parallel to v. From a practical point of view, the reconstruction problem is of particular interest. In the case that $m = 2$, the reconstruction of a lattice subset from line sums in m directions can be done in polynomial-time. For $m \geq 3$, the authors show that this problem is NP-hard.

The LCPMP can be used to model a special case of the reconstruction problem on the lattice $\mathbb{Z}^2$. In this special case, there are two directions $\binom{1}{0}$ and $\binom{0}{1}$. The lines parallel to these directions correspond to the rows and the columns of the 2-dimensional lattice. All line sums for lines in these directions, i.e. the sums of the values of points in any row or in any column of the lattice, are 1. Additionally, there is a single line sum for a third direction which adds up the values of the points on the main diagonal (in terms of a matrix) of the lattice. We briefly describe the construction of the graph of the LCPMP instance. Each node in one color class of the graph corresponds to one row of the lattice and each node in the other color class corresponds to one column of the lattice. Consequently, an edge $[i, j]$ in the graph corresponds to the point in row i and column j of the lattice (or vice versa, depending on the color classes i and j are in). One can see that when the nodes in the graph have appropriate indices, then the points on the main diagonal of the lattice are represented by the set of on-level edges in the graph. Hence, the level constraint models the line sum corresponding to the diagonal entries. In a perfect matching each node is covered by exactly one edge. This models the fact that the line sums corresponding to the first two directions are all 1. From a modeling point of view it is very restrictive to allow only instances with line sums of 1 for these directions, but at the same time the LCPMP allows for modeling that some points in the lattice must have a value of 0: This is due to the underlying graph, which does not need to be complete. Hence, all points corresponding to edges which are not in the graph are assumed to have a value of 0.

It is interesting to see that the LCPMP is a problem between the two cases $m = 2$ and $m = 3$, i.e. the polynomially solvable case and the case which is NP-hard. Complexity related investigations on the LCPMP are carried out in Chapter 4.

3. Complexity of Couple Constrained Matching

We have seen in Section 2.4.1 that the CCMP on bipartite graphs can be polynomially reduced to the maximum paired integer equal flow problem. This can be interpreted in the way that the bipartite CCMP is "not more difficult" than the maximum paired integer equal flow problem. Unfortunately, this does not imply complexity statements for the (bipartite) CCMP, as the maximum paired integer equal flow problem is NP-hard (compare Theorem 2.23) and thus belongs to the class of "hardest" solvable problems. So, the reduction does not answer the question of what complexity the CCMP is. In this chapter, we answer this question by presenting a new complexity result on this problem, which is that the CCMP is NP-hard, even on the class of bipartite cycles.

We start by giving a formulation of the problem *weighted matching with bonds*. Padberg and Sassano introduce this problem in [45]. They are the first who investigate the complexity of a matching problem additionally demanding the same matching activity for subsets of edges. Edges that are required to have the same matching activity are grouped in so-called bonds. The authors first consider the general case where bonds can be of any size, and show that this problem is NP-hard. Later, they transfer their complexity result to the case of bonds having cardinality at most 2. It is important to notice that – in contrast to the CCMP – weighted matching with bonds includes edge weights, which makes the problem more general than the CCMP. Originally, Padberg and Sassano simply refer to this problem as *matching with bonds*. We use the extended name to emphasize that it is a maximum weight problem.

To show that the CCMP is NP-hard, we extend Padberg's and Sassano's ideas from [45] while always considering the unweighted case. So, we first prove the new result that matching with bonds without edge weights is NP-hard. This is done by reduction from the independent set problem. Then, after showing some intermediate results, we prove that the CCMP is NP-hard. The complexity result we establish is even stronger: Along the reduction, we define the problem instances of the CCMP on cycle graphs only. Finally, this results in the fact that the CCMP is NP-hard even if the underlying graph is a bipartite cycle.

The chapter concludes with complexity results concerning the CCAP. For that, the special case where all couples contain only on-level edges is considered.

3.1. The complexity of matching with bonds

Padberg and Sassano introduce the notion of weighted matching with bonds in [45]. We follow their notation, which is given in the next definition.

Definition 3.1. *Let $G = (V, E)$ be a graph. A* bond structure *in G is a partition $\mathcal{B}$ of E. The sets $B \in \mathcal{B}$ are called* bonds.

Next, we present the formulation of the problem of finding a maximum weight matching such that either all edges of a bond are in the matching or none of them. Let $G = (V, E)$ be a graph and let $c_e \in \mathbb{R}$ be the weight of edge e for all $e \in E$. Further, let $\mathcal{B}$ be a bond structure in G.

Problem Formulation 3.2 (Weighted matching with bonds)**.** Weighted matching with bonds (w-MB) *is stated as the following integer linear program:*

$$\begin{aligned} \max \quad & \sum_{e \in E} c_e x_e && && (3.1) \\ s.t. \quad & \sum_{e \in \delta(v_i)} x_e \leq 1 && \forall\ i = 1, \dots, n && (3.2) \\ & x_e = x_f && \forall\ e, f \in B, \forall\ B \in \mathcal{B} \text{ with } |B| \geq 2 && (3.3) \\ & x_e \in \{0, 1\} && \forall\ e \in E. && (3.4) \end{aligned}$$

Constraints of the form (3.3) are called *bond constraints.* One should note that given a set of pairwise disjoint subsets of E, the set can always be completed to a bond structure by adding single-edge bonds to it; one for each edge that is not yet included in the given subsets.

Padberg and Sassano show that w-MB is NP-hard by reduction from the weighted independent set problem (which they refer to as vertex-packing). Furthermore, they show that the assertion stays true even when the underlying graph is restricted to be a simple cycle [45].

Theorem 3.3. *Weighted matching with bonds on cycles is NP-hard.*

We now show that this complexity result can be generalized to be valid also in the unweighted case. For that, we investigate the class of those problems among w-MB, where all edge weights are equal to 1. In the resulting class of problems, the task is to find a maximum cardinality matching in the graph G which satisfies the bond constraints. We prove that this class of problems is NP-hard.

Problem Formulation 3.4 (Matching with bonds)**.** Matching with bonds (MB) *is stated as the following integer linear program:*

$$\begin{aligned} \max \quad & \sum_{e \in E} x_e && (3.1') \\ s.t. \quad & (3.2) - (3.4). \end{aligned}$$

MB arises from w-MB by replacing the objective function (3.1) with (3.1'). This is a common specialization step in combinatorial optimization, as it equalizes the impact of the edges on the objective value and hence changes the problem type from maximum weight to maximum cardinality.

In order to show that MB is NP-hard, we now prove that the independent set problem is polynomially reducible to matching with bonds. The proof begins with constructing a simple cycle $\bar{G}$ and a bond structure $\bar{\mathcal{B}}$. This will be done by generalizing Padberg's and Sassano's proof of Theorem 3.3. Then, we extend the cycle graph and the bond structure in order to get a problem instance $(G^*, \mathcal{B}^*)$ of the unweighted problem MB. Beside constructing this MB problem instance explicitly, we will also point out which general properties the underlying graph needs to have in order to make the reduction work. One should note that we reduce from the unweighted independent set problem, in contrast to Padberg and Sassano, who reduce from the weighted independent set problem.

Theorem 3.5. *Matching with bonds is NP-hard.*

Proof. Suppose we are given an instance of the *independent set problem*, which is a connected graph $H = (V_H, E_H)$, with $V_H = \{v_i \mid i = 1, \ldots, |V_H|\}$. The independent set problem consists of finding a subset S of V_H which has maximum cardinality, such that no two nodes in S are connected by an edge in E_H.

Denote by $\pi = (v_{i_1}, e_{i_1}, \ldots, e_{i_m}, v_{i_{m+1}} = v_{i_1})$ an optimal *Chinese postman tour* of H, i.e. a closed walk of minimum cardinality which uses every edge of H at least once.

Based on π, we construct a graph $G^* = (V^*, E^*)$ and a bond structure $\mathcal{B}^*$ to get an instance $(G^*, \mathcal{B}^*)$ of MB, whose solution reveals a solution of the independent set problem on H. Finally, the graph G^* is a simple cycle together with some isolated, node-disjoint edges.

The intuition is as follows: Each edge in G^* represents a node in π and hence, also a node in V_H. A single node in V_H may be represented by several nodes in π. The edges in G^* representing the same node in V_H are accumulated in one bond. Hence, there is a one-to-one correspondence between the bonds in G^* and the nodes in V_H.

We describe $(G^*, \mathcal{B}^*)$ in two steps. First, we define a simple cycle $\bar{G} = (\bar{V}, \bar{E})$ on the edges of the Chinese postman tour. This cycle is a subgraph of G^*.

$$\begin{aligned}
\bar{V} &= \{e_{i_k} \mid k = 1, \ldots, m\}, \\
\bar{E} &= \{[e_{i_{k-1}}, e_{i_k}] \mid k = 2, \ldots, m+1\},
\end{aligned}$$

where $e_{i_{m+1}}$ is identical to e_{i_1}. The bond structure $\bar{\mathcal{B}}$, which partitions the edge set $\bar{E}$, is defined as

$$\bar{\mathcal{B}} = \{\bar{B}_i \mid i = 1, \ldots, |V_H|\}, \quad \text{with } \bar{B}_i = \{[e_{i_{k-1}}, e_{i_k}] \in \bar{E} \mid v_{i_k} = v_i\} \text{ for all } v_i \in V_H.$$

An example of $\bar{G}$ and $\bar{\mathcal{B}}$ can be found in Figure 3.3 (neglect isolated edges in part c). The structure of $\bar{G}$ and $\bar{\mathcal{B}}$ is as follows: Each node in $\bar{V}$ corresponds to an edge in π and for each occurrence of an edge e_i in π we have one node in $\bar{V}$. Two nodes $e_{i_{k-1}}, e_{i_k}$ in $\bar{V}$ are adjacent in $\bar{G}$ if and only if the sequence $(e_{i_{k-1}}, v_i, e_{i_k})$ is contained in π for some $i = 1, \ldots, |V_H|$. So, each edge in $\bar{E}$ corresponds to a node $v_i \in V_H$. Since nodes in V_H can be visited more than once in π, more than one edge in $\bar{E}$ can correspond to the same node $v_i \in V_H$.

The relationship between edges in $\bar{E}$ and nodes in V_H is used when composing the bonds. A bond $\bar{B}_i \in \bar{\mathcal{B}}$ contains an edge $[e_{i_{k-1}}, e_{i_k}] \in \bar{E}$ if and only if the sequence $(e_{i_{k-1}}, v_i, e_{i_k})$ is contained in π. So, each bond $\bar{B}_i \in \bar{\mathcal{B}}$ corresponds to a node $v_i \in V_H$. As a result, $\bar{G}$ is a simple cycle and $\bar{\mathcal{B}}$ is a bond structure on it.

Before we complete the description of G^* and $\mathcal{B}^*$, we show why our present graph and bond structure already guarantee a relation which is substantial for our reduction:

Claim. *A subset S of V_H is an independent set in H if and only if $M := \bigcup_{i:v_i \in S} \bar{B}_i$ is a matching in $\bar{G}$.*

Proof of Claim. We show that two nodes $v_p, v_q \in V_H$ are adjacent in H if and only if the corresponding edges in $\bar{B}_p$ and $\bar{B}_q$ cannot be together in a matching in $\bar{G}$. Let $v_p, v_q \in V_H$ be the two end-nodes of an edge $e_{i_k} \in E_H$. As π is a Chinese postman tour it does contain the edge e_{i_k} at least once, and there exist edges $e_{i_{k-1}}$ and $e_{i_{k+1}}$ in E_H so that π contains the sequence $(e_{i_{k-1}}, v_p, e_{i_k}, v_q, e_{i_{k+1}})$. For $\bar{G}$ this means that there is an edge in $\bar{B}_p$ and an edge in $\bar{B}_q$, both incident to node e_{i_k} (see Figure 3.1). These two edges cannot appear together in a matching in $\bar{G}$. Since the bond constraints state that either all or none of the edges in one bond shall be in a matching, the edges in $\bar{B}_p$ and $\bar{B}_q$ are not allowed to be together in the same matching.

Now let us assume that there are two edges $u \in \bar{B}_p$ and $v \in \bar{B}_q$, $p \neq q$, which are both incident to the same node $e_{i_k} \in \bar{V}$. We show that the nodes v_p and v_q are adjacent in H. As u and v are edges in the simple cycle $\bar{G}$, it contains the sequence $(e_{i_{k-1}}, u, e_{i_k}, v, e_{i_{k+1}})$ (see Figure 3.2). Since nodes in $\bar{G}$ are adjacent if and only if the corresponding edges appear consecutively in π, there exists a sequence $(e_{i_{k-1}}, v_p, e_{i_k}, v_q, e_{i_{k+1}})$ in π. Hence, v_p and v_q are adjacent in H. □

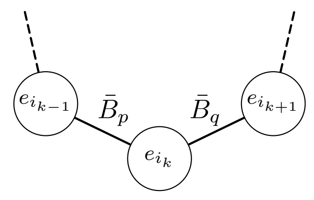


Figure 3.1.: Sequence with two edges in bonds $\bar{B}_p$ and $\bar{B}_q$, both incident to node e_{i_k}.

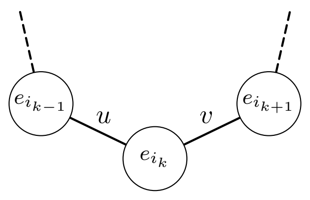


Figure 3.2.: Sequence with two edges u and v, both incident to node e_{i_k}.

Note that due to the bond constraints all matchings M in an MB problem are of the form $M = \bigcup_{i \in I,} \bar{B}_i$, for some index set I. For the corresponding independent set S, we have that $S = \{v_i \mid \bar{B}_i \subseteq M\}$.

Next, we finish the construction of G^* and the bond structure $\mathcal{B}^*$. Our aim is to have a bond structure with bonds of equal size. We make use of the fact that each occurrence of a node v_i in π correlates with one edge in the bond $\bar{B}_i$. For each node v_i in V_H, let c_{v_i} be the number of its occurrences in π. Further, let $c_{\max} := \max\{c_{v_i} \mid v_i \in V_H\}$. For each

$v_i \in V_H$ we define a set E_i, which consists of $c_{\max} - c_{v_i}$ pairwise node-disjoint edges:

$$E_i := \{[u_i^j, w_i^j] \mid j = 1, \ldots, c_{\max} - c_{v_i}\}, \tag{3.5}$$

with $u_i^j \neq w_i^j$ and $u_i^j, w_i^j \notin \bar{V}$ for all $i = 1, \ldots, |V_H|, j = 1, \ldots, c_{\max} - c_{v_i}$. If $c_{v_i} = c_{\max}$, we set E_i to be the empty set.

The graph G^* is composed of the simple cycle $\bar{G}$ and the edges defined in (3.5):

$$E^* := \bar{E} \cup \left(\bigcup_{i=1,\ldots,|V_H|} E_i \right).$$

We also add the new edges to the bonds and define $B_i^* := \bar{B}_i \cup E_i$ for all $i = 1, \ldots, |V_H|$. This completes the bond structure $\mathcal{B}^* = \{B_1^*, \ldots, B_{|V_H|}^*\}$. An example for G^* and $\mathcal{B}^*$ can be found in Figure 3.3. By augmenting the bonds this way, we ensure that $\mathcal{B}^*$ is a partition of E^* with all bonds being of the same size, i.e.

$$|B_1^*| = \ldots = |B_{|V_H|}^*| = c_{\max}.$$

This allows us to compare the size of an independent set S in H to the size of the corresponding edges $M = \bigcup_{i:v_i \in S} B_i^*$ in G^*:

$$|S| = |\{i \mid B_i^* \subseteq M\}| = \frac{1}{c_{\max}} |M|.$$

As a consequence, $S \subseteq V_H$ is an independent set of maximum size in H if and only if $M = \bigcup_{i:v_i \in S} B_i^*$ is a matching of maximum size in G^*.

The reduction described above can be done in polynomial time. Edmonds and Johnson describe how to find an optimal Chinese postman tour π in polynomial time [22]. They also show that every edge in H is used at most twice by π. So, in π each node $v_i \in V_H$ is visited at most $\frac{2\deg(v_i)}{2} = \deg(v_i)$ times, where $\deg(v_i)$ is the degree of node v_i for all $v_i \in V_H$. As a consequence, $c_{\max}$ is bounded above by the maximum degree $\deg_{\max}$ in H. This leads to

$$|V^*| \leq 2|E_H| + 2\sum_{i=1}^{|V_H|} |E_i| \leq 2|E_H| + 2c_{\max}|V_H| \leq 2|E_H| + 2\deg_{\max}|V_H|,$$

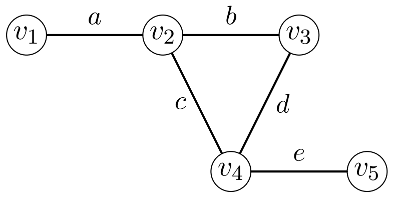
$$|E^*| = |\bigcup_{i=1}^{|V_H|} B_i^*| = c_{\max}|V_H| \leq \deg_{\max}|V_H|.$$

This completes the proof. □

In the preceding proof an instance of MB consisting of a graph and a bond structure was constructed. We have seen that the graph of this problem instance has a maximum node degree of 2. Corollary 3.6 follows as a direct consequence of this fact.

Corollary 3.6. *Matching with bonds on graphs with maximum node degree 2 is NP-hard.*

a)



b) $\pi = (v_1, a, v_2, c, v_4, e, v_5, e', v_4, d, v_3, b, v_2, a', v_1)$

c)

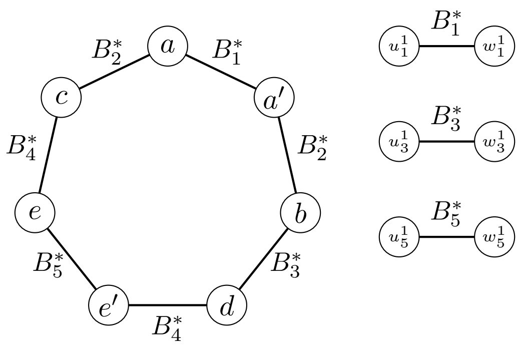


Figure 3.3.: a) An example graph H. Nodes v_1, v_3, v_5 form a unique maximum independent set. b) Optimal Chinese postman tour π in H. Dashes indicate a second visit to edges. c) Graph G^* for problem MB. Bonds are denoted along the edges they contain.

3.2. The complexity of the couple constrained matching problem

We now switch from bonds to couples and come from matching with bonds to the couple constrained matching problem. In order to prove that the CCMP is NP-hard, some intermediate complexity results will be established first. This will be done by showing a sequence of problem reductions. It can be summarized as follows.

We begin with the *node cover* problem (NC) on graphs with a node degree of at most 3. This problem is polynomially reduced to NC on 4-regular graphs. The node cover problem and the independent set problem have equivalent complexity, as the complement of a node cover is an independent set, and vice versa (see [26] for a detailed proof). Making use of this fact, we switch from NC to the independent set problem and continue by polynomially reducing the independent set problem on 4-regular graphs to MB with bond size 2. Finally, this problem will be polynomially reduced to the CCMP. The sequence of problem reductions is depicted in Figure 3.4.

The independent set problem, MB and the CCMP will be treated as optimization problems. Instead of considering NC as an optimization problem, we will consider the decision version of NC. As both problems are polynomially equivalent [26], switching from optimization to decision problems does not affect the NP-hardness results.

Concerning the first problem reduction, Garey et al. prove that the NC decision problem on graphs with node degree at most 3 is NP-complete [27], [26]. As they do not show NP-completeness of the problem restricted to 4-regular graphs explicitly, this is what we show next.

Lemma 3.7. *The node cover decision problem on 4-regular graphs is NP-complete.*

Proof. It is obvious that the NC decision problem on 4-regular graphs is in NP, as the same problem without restrictions on the graph is already in NP.

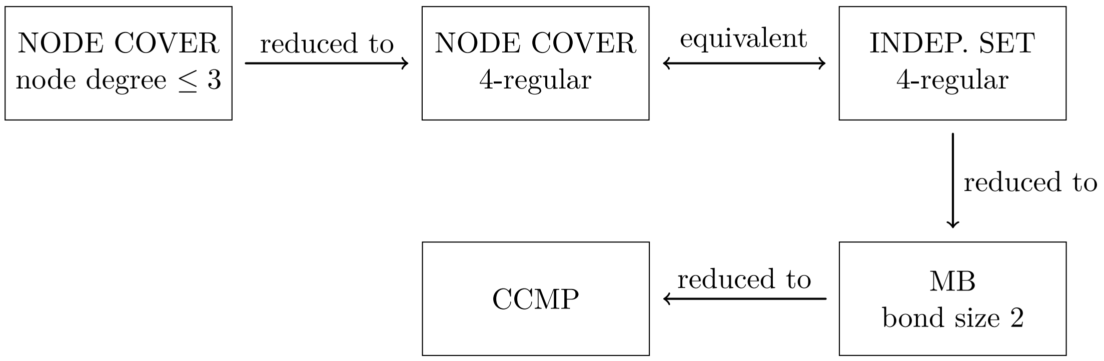


Figure 3.4.: Sequence of problem reductions

So, it remains to be shown that our variant of the problem is NP-hard. We make use of the fact that the NC decision problem on graphs with node degree at most 3 is NP-complete (see [27], [26]). In the following it is described how this problem can be polynomially reduced to the NC decision problem on 4-regular graphs.

Let $(H = (V_H, E_H), k)$ with $V_H = \{v_1, \ldots, v_{|V_H|}\}$ be an instance of the NC decision problem on graphs with node degree at most 3. The question to answer is whether H contains a node cover $S \subseteq V_H$ of size less than or equal k. We will reduce (H, k) to a problem instance of the NC decision problem on a graph $G' = (V, E)$ with an integer k', which are both defined based on the given problem instance (H, k). As required, G' will be a 4-regular graph.

The graph H will serve as basis for the construction of the graph G'. We denote the sum of all node degrees in H by z_H, i.e. $z_H := \sum_{i=1,\ldots,|V_H|} \deg(v_i)$. Of course, z_H is an even integer number. Further, let $d_i := 4 - \deg(v_i)$ be the *degree deficit* of node $v_i \in V_H$ for all $i = 1, \ldots, |V_H|$. The degree deficit of a node in H denotes how many edges (non-loops) incident to that node are missing, to be a node of degree 4. The total degree deficit of H is denoted by d_H, so we have

$$d_H = \sum_{v_i \in V_H} d_i = 4|V_H| - z_H.$$

We ensure to get a 4-regular graph by "filling up" node degrees in H. This is done with the aid of special structured subgraphs we call *bricks*. A brick is a graph $B = (V_B, E_B)$ which is constructed as follows (see Figure 3.5):

$$V_B = \{b_i \mid i = 1, \ldots, 6\},$$
$$E_B = \{[b_i, b_j] \mid 1 \leq i < j \leq 5 \text{ with } i \neq 4\} \cup \{[b_4, b_6], [b_5, b_6]\}.$$

In other words, nodes $b_1, \ldots, b_5$ in the brick induce the graph $K_5 - [b_4, b_5]$. Instead of the edge $[b_4, b_5]$, B contains the edges $[b_4, b_6]$ and $[b_5, b_6]$. One can see directly that every node in B has degree 4, except for node v_6, which has degree 2.

Brick graphs as defined above are employed in the following way: A brick B is inserted in the graph H by connecting the brick node v_6 to two nodes v_i, v_j in V_H with $\deg(v_i) < 4$

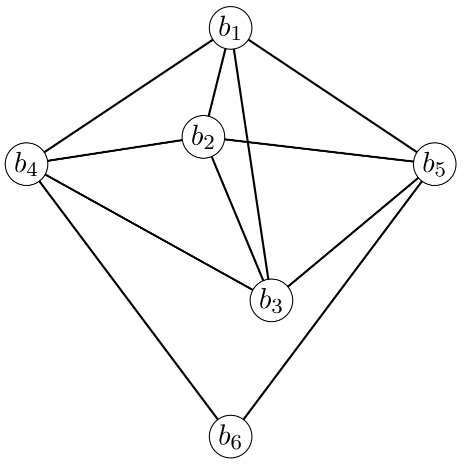


Figure 3.5.: Brick B

and $\deg(v_j) < 4$. As a result, node b_6 has degree 4, and the total degree deficit of the resulting graph $H + B$ is decremented to $d_{H+B} = d_H - 2$. We call the new edges $[b_6, v_i]$ and $[b_6, v_j]$ *docking edges*.

The graph G' is constructed by repeating the above procedure until the total degree deficit of the resulting graph is 0, which means that it is 4-regular. One should note that it is always possible to get a 4-regular graph G' that way, since the total degree deficit starts from d_H, which is an even number, and decreases by 2 each time a brick is added. In order to distinguish the bricks in G' from each other, let $B_1, \dots, B_p$ with $p = \frac{|d_H|}{2}$ be the brick subgraphs which are added to H. The nodes in a brick B_i are denoted by $b_1^i, \dots, b_6^i$.

The second part of the problem instance is the parameter k', which is defined as

$$k' := k + 4p.$$

We will show now that (H, k) is a yes-instance of the NC decision problem if and only if (G', k') is a yes-instance of the NC decision problem. First, we consider an isolated brick B. One can see (e.g. by enumeration) that a minimum node cover in B consists of 4 nodes. Further, $S_B := \{b_1, b_2, b_3, b_6\}$ is a minimum node cover in B which contains the node b_6.

Now, let (H, k) be a yes-instance of the NC decision problem. This means there is a node cover $S_H \subseteq V_H$ of cardinality $|S_H| \leq k$. We define

$$S' := S_H \cup \left(\bigcup_{i=1}^{p} S_{B_i} \right),$$

with $S_{B_i} = \{b_1^i, b_2^i, b_3^i, b_6^i\}$ for all $i = 1, \dots, p$, corresponding to S_B as defined above.

The set of edges in G' can be partitioned into the sets

$$E_H, E(B_i), E(D_i), \quad \text{for } i = 1, \dots, p,$$

where $E(B_i)$ and $E(D_i)$ are the set of all edges in brick B_i and the set of the two docking edges emanating from brick B_i, respectively, for all $i = 1, \dots, p$. Since S_H is a node cover in H, the edges in E_H are covered by S'. For any $i = 1, \dots, p$ we have that S_{B_i} is a node cover in B_i which contains the node b_6^i. So, S' covers all edges in $E(B_i)$ and both docking edges in $E(D_i)$, as they are both incident to node b_6^i. Concerning the size of S' we have that $|S'| = |S_H| + 4p \leq k + 4p$. As a result, S' is a node cover in G' with cardinality $|S'| \leq k'$.

Next, let (G', k') be a yes-instance of the NC decision problem. Hence, there exists a node cover $S' \subseteq V$, with $|S'| \leq k'$. We have seen that each node cover contains at least 4 nodes in every brick. As there are p brick subgraphs in G', it follows that at most $|S'| - 4p$ nodes in V_H suffice to cover all edges in E_H. The term $|S'| - 4p$ can be estimated by

$$|S'| - 4p \leq k' - 4p = k,$$

which shows that (H, k) is a yes-instance of the NC decision problem.

What is left to show is that the problem transformation as described above is polynomial in the input size of (H, k). Determining the degree of each node in H can surely be done in polynomial time. Consequently, the calculation of the problem parameter k' can also be done in polynomial time. G' comprises the original graph H and p bricks plus two docking edges per brick. As each brick subgraph is of the same constant size, the total size of all bricks including docking edges in G' is in $\mathcal{O}(p) = \mathcal{O}\left(\frac{|d_H|}{2}\right) = \mathcal{O}(V_H)$. It is possible to determine which nodes get connected to bricks in the following way. All nodes in H and their degrees are maintained in a list L. Two successive nodes in L with positive degree deficit get connected to a brick. This is done until all degree deficits are 0. As the maximum degree deficit of a node is 4, the number of steps needed for this procedure is polynomially bounded by the size of the list, which is $|V_H|$. □

The complexity result of Lemma 3.7 can be used directly to strengthen the statement of Theorem 3.5. For that, the independent set problem on 4-regular graphs is polynomially reduced to MB.

Lemma 3.8. *Matching with bonds on cycles is NP-hard, even if all bonds have cardinality 2.*

Proof. To show this result, we consider the proof of Theorem 3.5. Now, let $H = (V_H, E_H)$ be an instance of the independent set problem, with H being a 4-regular graph.

The key property of H is that it is an Eulerian graph, which means that there exists a tour in H which contains every edge in E_H exactly once. Hence, this tour is also an optimal Chinese postman tour π in H. As the underlying graph H is 4-regular, each node $v_i \in V_H$ is visited exactly twice in π. Following the proof of Theorem 3.5, we use π to get a simple cycle $\bar{G}$ and a bond structure $\bar{\mathcal{B}}$. Due to the fact that each node $v_i \in V_H$ is visited exactly twice in π, we have that each bond $B_i \in \bar{\mathcal{B}}, i = 1 \dots, |V_H|$, consists of exactly 2 edges. Hence, all bonds in $\bar{\mathcal{B}}$ are of the same size already and there is no need for additional edges to adjust the difference in the bonds' cardinalities. Therefore,

$G^* := \bar{G}$ and $\mathcal{B}^* := \bar{\mathcal{B}}$ are the final graph and bond structure and form a problem instance of MB.

As we applied the same construction techniques as in the proof of Theorem 3.5, we note that G^* is a simple cycle and all bonds in $\mathcal{B}^*$ have cardinality 2. □

As a result of Lemma 3.8, MB stays NP-hard even under the restrictions that the underlying graph is a cycle and all bonds are of size 2. We keep these restrictions on MB and finally show that this problem is polynomially reducible to the CCMP.

First, the difference between MB and the CCMP is clarified. Instead of a bond structure, the CCMP has a couple collection $\mathcal{F} = \{F_1, \ldots, F_k\}$. All couples in $\mathcal{F}$ must be of size 2. A couple collection – in contrast to a bond structure – does not need to be a partition of the set of edges of the underlying graph, but it can. This implies that MB on cycles with bonds having cardinality 2 is a special case of the CCMP on cycles.

Theorem 3.9. *The couple constrained matching problem is NP-hard on cycles.*

Proof. We show that MB on cycles with bonds having cardinality 2 is polynomially reducible to the CCMP on cycles. Let $(G, \mathcal{B})$ be a problem instance of MB on cycles with bonds having cardinality 2. As all of the bonds are of cardinality 2, they can be simply renamed couples. Thus, $\mathcal{F} := \mathcal{B}$ is a couple collection with $F_i := B_i$ for all $B_i \in \mathcal{B}$. Now, $(G, \mathcal{F})$ is a problem instance of the CCMP and it has the same optimal solution as the problem instance $(G, \mathcal{B})$ of MB. As Lemma 3.8 states that MB on cycles with bonds of cardinality 2 is NP-hard, it follows that CCMP on cycles is NP-hard. □

A final remark on MB with restrictions should be mentioned here. In case the restrictions on the bond sizes as given in Lemma 3.8 are loosened to be of the form $|B| \leq 2$ for all $B \in \mathcal{B}$, a stronger relationship between MB and the CCMP holds: MB with bonds having cardinality at most 2 is polynomially equivalent to the CCMP.

3.2.1. The bipartite case

In the previous section we showed that the couple constrained matching problem is NP-hard on cycle graphs (see Theorem 3.9). This result was established with the help of Lemma 3.8. There is a further implication from that lemma, which allows us to state that the couple constrained matching problem is NP-hard even on bipartite cycles. We make use of the fact that bonds partition the edge set of a graph. As the underlying graph is a cycle and all bonds are of cardinality 2, the cycle is of even length and thus, it is bipartite. In the reduction we use in the proof of Theorem 3.9, MB on cycles with bonds having cardinality 2 is reduced to the CCMP without changing the underlying graph. This yields the following result.

Corollary 3.10. *The couple constrained matching problem is NP-hard on bipartite cycles.*

The previous result can also be shown directly by reducing the CCMP on cycles to the CCMP on bipartite cycles. We show this reduction next.

Theorem 3.11. *The couple constrained matching problem on cycles can be polynomially reduced to the couple constrained matching problem on bipartite cycles.*

Proof. Let $(G, \mathcal{C})$ be an instance of the CCMP, where $G = (V, E)$ is a cycle and $\mathcal{C} = \{C_1, \dots, C_p\}$ is a couple collection on G. Let k denote the length of the cycle G and let $V = \{v_1, \dots, v_k\}$ be an ordering of the nodes corresponding to their sequential appearance in the cycle, i.e. $[v_i, v_{i+1}] \in E$ for all $i = 1, \dots, k$, with v_{k+1} being identical to v_1. Hence, G can be written as $[v_1, e_1, v_2, \dots, v_k, e_k, v_1]$.

In the case that k is even, G is bipartite and no problem reduction needs to be done. In the case that k is odd, we transform the problem instance $(G, \mathcal{C})$ into the problem instance $(H, \mathcal{D})$, with $H = (W, F)$ being a bipartite cycle graph and $\mathcal{D}$ being a couple collection on H. For that, let $b \in E$ denote an edge in G that is not contained in any couple in $\mathcal{C}$. Such an edge exists, since G contains an odd number of edges and the number of edges of all the couples in $\mathcal{C}$ is even. Without loss of generality, let $b = e_k = [v_k, v_1]$.

Loosely spoken, the graph H corresponds to a doubling of the cycle G, where the two resulting cycles are connected by a modified edge b and its duplicate. Figure 3.6 depicts the construction scheme of the graph H and the couple collection $\mathcal{D}$. More formally, let the set of nodes of H be defined as $W := V \cup \{v'_i \mid i = 1, \dots, k\}$. The edge set of H contains all the edges $f_i := e_i = [v_i, v_{i+1}]$ which also appear in $E \setminus \{b\}$, and their corresponding duplicates $f'_i := [v'_i, v'_{i+1}]$. So, $F := \{f_i, f'_i \mid e_i \in E, i = 1, \dots, k-1\}$. So far, H consists of two broken-up cycles. A connection between them is achieved by adding two more edges to F:

$$f_k := [v_k, v'_1] \text{ and } f'_k := [v'_k, v_1].$$

Now, H is an even cycle of the form

$$[v_1, f_1, v_2, \dots, v_k, f_k, v'_1, f'_1, v'_2, \dots, v'_k, f'_k, v_1].$$

For each couple $C_s = \{e_i, e_j\}$ in $\mathcal{C}$ there exists a couple $D_s = \{f_i, f_j\}$ and a couple $D'_s = \{f'_i, f'_j\}$. The two edges f_k and f'_k constitute an additional couple D^*. So, $\mathcal{D} = \{D_s, D'_s \mid s = 1, \dots, p\} \cup \{D^*\}$. This completes the description of the problem instance $(H, \mathcal{D})$.

One should note the "symmetry" of the two almost-cycles the graph H is composed of. It is reflected in the relationship between the edges f_i and f'_i. For any $i = 1, \dots, k$ it holds that f_i is incident to f_j if and only if f'_i is incident to f'_j. Concerning their occurrence in couples, it holds that an edge f_i is in the couple D_s if and only if its corresponding duplicate f'_i is in the couple D'_s. The couple D^* ensures that if f_k is contained in a solution matching N in H, then neither f_1 nor f_{k-1} are in N. Analogously, if f'_k is in a solution matching N in H, then neither f'_1 nor f'_{k-1} are in N. This reflects the incidence relationships of the corresponding edges in the original graph G.

Let N be a maximum matching in H which fulfills the couple constraints concerning $\mathcal{D}$. Due to the symmetry in H, the subsets $N_1 = \{f_i \in N \mid i = 1, \dots, k\}$ and $N_2 = \{f'_i \in N \mid i = 1, \dots, k\}$ are of the same size. Otherwise, one could discard the

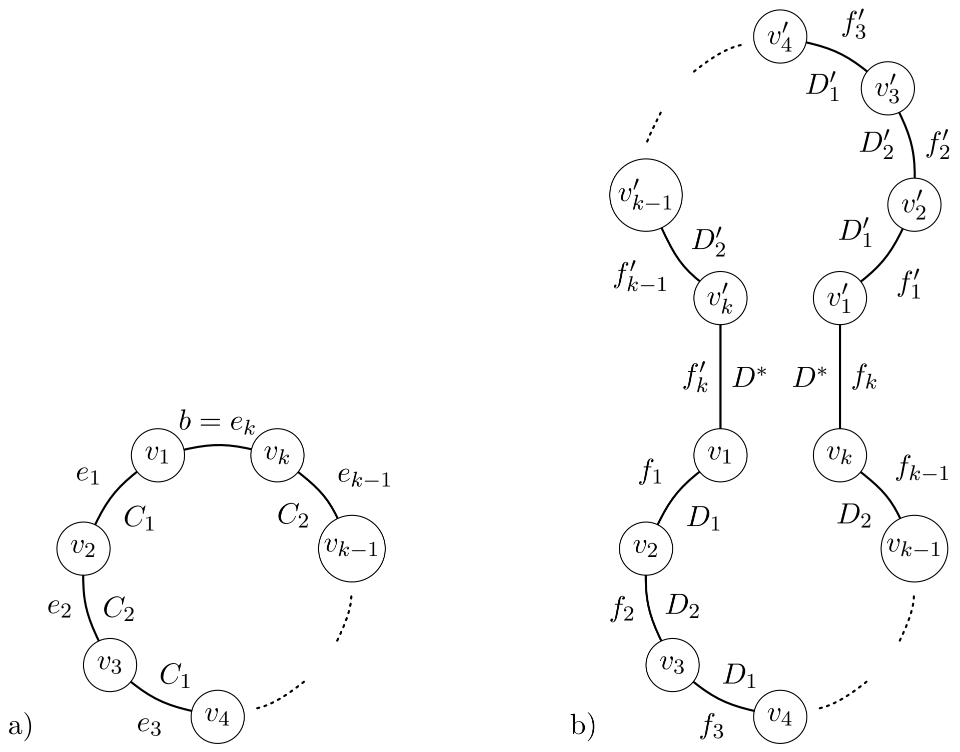


Figure 3.6.: a) Cycle graph of odd length k. Edge names e_i and their membership to example couples C_i are written along the edges, on the outer and inner side of the cycle, respectively. b) Even cycle graph constructed in problem reduction.

edges of the smaller of the two subsets and replace them with the symmetric counterpart of the larger one. To show that such a replacement would give a feasible matching, suppose that N_2 is the larger of the two subsets. Let $\bar{N}$ be the matching resulting from replacing N_1 in N with $\{f_i \mid f'_i \in N_2\}$. Obviously, $\bar{N}$ fulfills the couple constraints concerning $\mathcal{D}$. In order to be a feasible matching, neither the two edges f_1 and f'_k nor the two edges f_k and f'_1 may appear together in $\bar{N}$. We have already stated that the edges f'_k and f'_1 cannot be together in N. After the replacement, f_1 is in $\bar{N}$ if and only if f'_1 is in N. Hence, f_1 and f'_k are not both in $\bar{N}$. Analogously, f_k is in $\bar{N}$ if and only if f'_k is in N. Hence, f_k and f'_1 are not both in $\bar{N}$.

The solutions to the problem instances $(G, \mathcal{C})$ and $(H, \mathcal{D})$ are related in the following way. A matching M in G can be deduced from a matching N in H by the following rule:

$$M := \{e_i \in E \mid f_i \in N\}.$$

Obviously, M is a matching in G and when N satisfies the couple constraints concerning $\mathcal{D}$, then M satisfies the couple constraints concerning $\mathcal{C}$.

A consequence of the fact that both subsets N_1 and N_2 are of the same size is that

$$|M| = \frac{1}{2}|N|.$$

For the rest of this proof, we implicitly expect all mentioned matchings to fulfill the corresponding couple constraints. Now, we show that a maximum matching M in G is of size z if and only if a maximum matching N in H is of size $2z$.

Let M be a maximum matching in G of size z and assume there is a matching $\bar{N}$ in H with $|\bar{N}| > 2z$. Constructing $\bar{M}$ from $\bar{N}$ as described above yields a matching in G with $|\bar{M}| = \frac{1}{2}|\bar{N}| > z = |M|$, contradicting the optimality of M. Next, let N be a maximum matching in H of size $2z$ and assume that there is a matching $\bar{M}$ in G with $|\bar{M}| > z$. Then, $\bar{N} := \{f_i, f_i' \in F \mid e_i \in \bar{M}\}$ is a matching in H of size $|\bar{N}| = 2|\bar{M}| > 2z = |N|$, contradicting the optimality of N.

The graph H contains exactly twice as many nodes and edges as the graph G. The number of couples in $\mathcal{D}$ is $|\mathcal{D}| = 2|\mathcal{C}|+1$. Hence, the problem reduction is polynomial. □

The fact that we can restrict the underlying graph of the CCMP to bipartite cycles without changing the problem complexity opens up a link to level graphs, which are defined in Definition 2.8 in Section 2.2.

Corollary 3.12. *The couple constrained matching problem is NP-hard on level graphs.*

Proof. Due to Theorem 3.11, the CCMP is NP-hard on bipartite cycles. The nodes of any bipartite cycle $G = (U \mathbin{\dot\cup} V, E)$ with $U = \{u_1, \dots, u_n\}$ and $V = \{v_1, \dots, v_n\}$ can be renamed such that the set of edges is

$$E = \{[u_i, v_i] \mid i = 1, \dots, n\} \cup \{[u_i, v_{i+1}] \mid i = 1, \dots, n\},$$

where v_{n+1} is identical to v_1. The edge set E contains all on-level edges $[u_i, v_i]$, where $i = 1, \dots, n$. So, the bipartite cycle G is a level graph. Hence, the CCMP is NP-hard also when restricting the underlying graph to be a level graph. □

3.3. Complexity issues of the couple constrained assignment problem

The couple constrained assignment problem as defined in Section 2.1.3 corresponds to a minimum cost couple constrained matching problem on a complete bipartite graph.

Aboudi and Nemhauser [3] introduce the CCAP with an additional restriction on the couple collection. They assume a couple collection $\mathcal{F} = \{F_1, \dots, F_p\}$ to have the property that each node in the graph is incident to at most one edge in $\bigcup_{i=1}^{p} F_i$. Then, by renaming the nodes in the graph, the couple constraints (2.8) can be replaced with constraints of the form

$$x_{2k-1,2k-1} = x_{2k,2k} \quad \text{for all } k = 1, \dots, p. \tag{3.6}$$

Problem Formulation 3.13 (Couple constrained assignment with on-level couples)**.** *The* couple constrained assignment problem with on-level couples (CCAP-L) *is the couple constrained assignment problem where all couple constraints are of the form (3.6).*

In order to state the complexity of this problem, we take a further look at the couple constrained matching problem and recall the weighted version of it (compare Problem Formulation 2.25): The *weighted couple constrained matching problem (w-CCMP)* is the problem of finding a maximum weight matching satisfying the couple constraints.

As stated in Section 3.2, the CCMP is NP-hard on bipartite cycle graphs. Thus, the w-CCMP on bipartite cycle graphs is NP-hard as well. We show that this result holds when assuming that no two edges out of any couples share an end-node. Then, a reduction to the CCAP-L is easy.

Lemma 3.14. *The weighted couple constrained matching problem on bipartite cycles, where no two edges of any couples share an end-node, is NP-hard.*

Proof. We polynomially reduce the w-CCMP on bipartite graphs to the w-CCMP on bipartite cycles where no two edges in any couple share an end-node.

Let $(G, c, \mathcal{F})$ be an instance of the w-CCMP, where $G = (U \mathbin{\dot\cup} V, E)$ is a bipartite cycle graph with $U = \{u_1, \dots, u_n\}, V = \{v_1, \dots, v_n\}$ and edge weights $c_e \in \mathbb{R}$ for all $e \in E$. Let $\mathcal{F} = \{F_1, \dots, F_p\}$ be a couple collection for which we assume that among the edges of the couples there exist at least two edges which share an end-node. Clearly, as G is a cycle, there cannot be more than 2 edges sharing the same end-node. The number of nodes in G which appear to be end-nodes of two edges of any couples is denoted by r. Let M be defined as an integer that is bigger than the sum of the absolute values of all edge weights: $M := \sum_{e \in E} |c_e| + 1$.

Now, for each two edges $e_s = [u_h, v_i] \in F_s$ and $e_t = [v_i, u_j] \in F_t$, $s \neq t$, which share an end-node v_i, we put two additional edges f_i, g_i in between them as follows. We replace the path $[u_h, v_i, u_j]$ with $[u_h, v_i', z_i, v_i'', u_j]$, as depicted in Figure 3.7. The edges e_s and e_t now have the end-nodes u_h, v_i' and v_i'', u_j, respectively. The new edges $f_i = [v_i', z_i]$ and $g_i = [z_i, v_i'']$ are assigned a weight of M, where all other edges keep their original weights. The resulting graph is denoted by H. One should note that the couple collection $\mathcal{F}$ has not been modified and is applied in the couple constraints in both graphs G and H.

Now, it holds that G contains a matching satisfying the couple constraints which is of weight l if and only if H contains a matching satisfying the couple constraints which is of weight $rM + l$. To see this, we first consider a matching N_G in G which satisfies the couple constraints and has weight l. Applying N_G to the graph H gives a matching N_H which also satisfies the couple constraints and has weight l. Additionally, for each pair of edges (f_i, g_i) in H we can add either f_i or g_i to N_H. This is possible, because at most one of the edges e_s and e_t is in N_H, due to their incidence in G. As there are r of such edge pairs, the resulting matching is of size $rM + l$.

Now we consider the remaining implication. Let N_H be a matching in H which satisfies the couple constraints and has weight $rM + l$. Then, N_H contains exactly one edge in each of the r pairs (f_i, g_i). This ensures that at most one of the edges e_s and e_t – which share an end-node in G – is in N_H. Removing all those edges which are of the form f_i

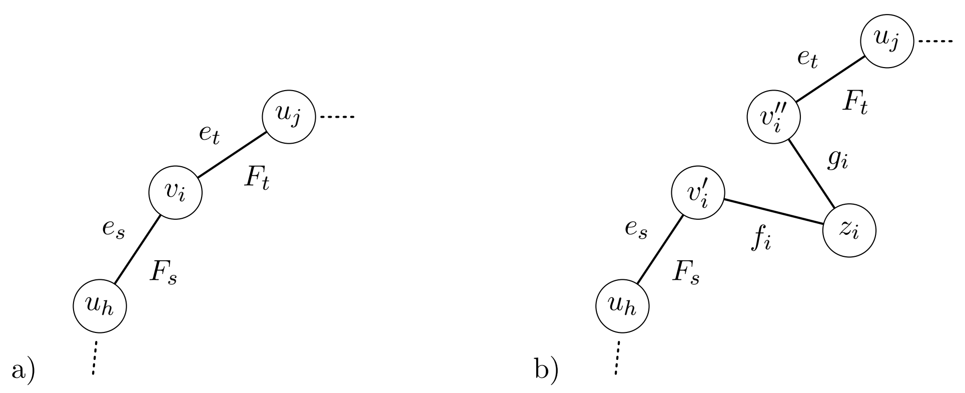


Figure 3.7.: a) Path with two edges e_s and e_t in couples F_s and F_t. Both edges share the end-node v_i. b) New path with two additional edges f_i and g_i. Here, e_s and e_t do not have an end-node in common.

or g_i from N_H yields a matching in G which still satisfies the couple constraints and has weight l.

The graph H built along this problem reduction is a bipartite cycle, as the added paths are of even length and only lengthen the cycle G. The reduction itself is polynomial, as the graph H contains at most $3|U \cup V|$ nodes and at most $|E| + 2|U \cup V|$ edges. □

With the help of Lemma 3.14 the complexity of the CCAP-L can be stated now:

Corollary 3.15. *The couple constrained assignment problem with on-level couples is NP-hard.*

Proof. Let $(G = (U \cupdot V, E), c, \mathcal{F})$ be a problem instance of the w-CCMP, where G is a bipartite cycle with $|U| = |V| =: n$ and $c_e \in \mathbb{R}$ is the weight of the edge e for all $e \in E$. Further, let the couple collection $\mathcal{F}$ be such that no two edges of any couples share an end-node. Without loss of generality, let the nodes be named such that the couples contain on-level edges only.

We show that each such problem instance can be solved by solving a problem instance $(K_{n,n} = (U \cupdot V, E'), c', \mathcal{F}')$ of the CCAP-L. For that, in the CCAP-L instance the costs of the edges of the graph $K_{n,n}$ are defined as follows. Each edge $e \in E'$ which is also in E and is not contained in any couple and has a nonnegative weight c_e is assigned the cost $c'_e := -c_e$. Each edge $e \in E'$ which is also in E and is contained in a couple $\{e, f\}$ with $c_e + c_f \geq 0$ is also assigned the cost $c'_e := -c_e$. All other edges in E' are assigned a cost of M, with $M := \sum_{e \in E} |c_e| + 1$. The couple collection in the CCAP-L instance is defined to be $\mathcal{F}' := \mathcal{F}$.

Then, the w-CCMP instance has an optimal solution N^* of weight z^* if and only if the CCAP-L instance has an optimal solution of cost $-z^* + tM$, where $t = n - |N^*|$.

As Lemma 3.14 states that the w-CCMP on bipartite cycle graphs with no two edges in any couples sharing an end-node is NP-hard, this simple problem reduction shows that the CCAP-L is NP-hard as well. □

One should note that the complexity of the CCAP-L depends on the costs in the objective function. When all costs are equal, the problem is easy to solve. Then, an optimal solution x^* is given by

$$\begin{aligned} x^*_{ii} &= 1 \quad \text{for all } i = 1, \dots, n, \\ x^*_{ij} &= 0 \quad \text{for all } i, j = 1, \dots, n \text{ with } i \neq j. \end{aligned}$$

The following lemma summarizes this complexity result:

Lemma 3.16. *The couple constrained assignment problem with on-level couples and uniform costs is polynomially solvable.*

4. Complexity Aspects of Level Constrained Matching

In this chapter, we discuss complexity aspects of the level constrained matching problem and related problems. The first part of this chapter deals with two perfect matching problems with different additional side constraints and a cycle packing problem with an additional side constraint. We show that each of these problems is polynomially equivalent to the LCMP (we do so by considering the LCPMP instead of the LCMP). One of the matching problems with additional side constraints considered is the equality constrained perfect matching problem. Alfakih et al. [7], Yi et al. [52] and Costa et al. [16] mention that it is an open question whether a polynomial algorithm exists for this problem. Due to its equivalence to the LCPMP, this increases the relevance of complexity results related to the LCPMP.

In the second part of this chapter, we take a closer look at the so-called restricted perfect matching problem, which is also polynomially equivalent to the LCPMP. We present changes in its problem formulation which are sufficient for the problem to become polynomially solvable or to become NP-hard. To this aim we investigate the impact of fixed and variable parameters in the additional side constraints to the complexity of the problem. We consider different combinations of fixed and variable parameters and present special cases from the literature. A final comparison of the different cases shows assumptions in the problem formulation which are of great importance for figuring out the complexity of the problem. This gives us an intuition about the complexity of this class of problems. We will see that the complexity of restricted matching problems (and thus, also the LCMP) lies on the edge between being NP-hard and polynomially solvable.

The third part of this chapter deals with the complexity of the assignment problem with an additional equality constraint. First, the case where the equality constraint affects *all* edges of the underlying complete bipartite graph is presented. Then, we discuss the case where the equality constraint affects an arbitrary subset of the edges.

For the rest of this chapter, we follow the widely assumed hypothesis that P $\neq$ NP. Hence, we assume that polynomially solvable problems and NP-hard problems are of different complexity.

4.1. Problems of the same complexity as the level constrained perfect matching problem

We will consider three problems for which we will show that they are of the same complexity as the LCMP. The problems are briefly described next. We remark that we consider the following problems as *perfect* matching problems and compare them to the LCPMP instead of the LCMP. This matches the problem definitions in the literature. Due to Theorem 2.12 this can be done without loss of generality, as complexity statements for the LCPMP also apply for the LCMP, and vice versa.

Problem Formulation 4.1 (Equality constrained perfect matching)**.** *Let $G = (U \cupdot V, E)$ be a bipartite graph with $U = \{u_1, \dots, u_n\}, V = \{v_1, \dots, v_n\}$. Let $R \subseteq E$ and let k be an integer-valued parameter with $0 \leq k \leq n$. The* equality constrained perfect matching problem *(ECPMP) is the problem of finding a perfect matching M in G such that*

$$|M \cap R| = k. \tag{4.1}$$

Problem Formulation 4.2 (Restricted perfect matching with fixed number of restrictions)**.** *Let $G = (U \cupdot V, E)$ be a bipartite graph with $U = \{u_1, \dots, u_n\}, V = \{v_1, \dots, v_n\}$. Let $R_1, \dots, R_l \subseteq E$ with l fixed. Further, let $r_1, \dots, r_l$ be positive integer values. The* restricted perfect matching problem with fixed number of restrictions *(l-RPMP) is the problem of finding a perfect matching M in G such that*

$$|M \cap R_i| \leq r_i \quad \text{for all } i = 1, \dots, l. \tag{4.2}$$

Problem Formulation 4.3 (Exact cycle sum)**.** *Let $D = (V, A)$ be a directed graph with $V = \{v_1, \dots, v_n\}$. Let k be an integer-valued parameter with $0 \leq k \leq n$. The* exact cycle sum problem *is the problem of finding a set of node-disjoint cycles of total length exactly k in D.*

The ECPMP is – just like the LCPMP – a perfect matching problem with an additional equality constraint. The set of edges this constraint refers to is not restricted to the set of on-level edges but rather to an arbitrary subset of the edge set. Therefore, it is a generalization of the LCPMP. The l-RPMP is also a perfect matching problem, but with a fixed number of additional inequality constraints. Its constraints define upper bounds on the number of edges of arbitrary subsets in a feasible matching. Each single constraint is less restrictive than the additional equality constraint of the ECPMP, but there can be more than one of them. The exact cycle sum problem is the problem of finding node-disjoint cycles in a directed graph, with one additional side constraint requiring that the sum of the lengths of all cycles equals a given value. This problem is not defined in terms of a perfect matching problem, but we will show in which way it is related to them.

Papadimitriou and Yannakakis [46] were the first who related the complexity of the above problems to each other. Originally, they formulated the ECPMP and the l-RPMP as *complete* matching problems with the corresponding side constraints. As we consider the case where both color classes of the underlying bipartite graph are of the same size, we formulate the problems as perfect matching problems. The following result is due to Papadimitriou and Yannakakis [46].

Theorem 4.4. *For each $l \geq 2$ the ECPMP, the l-RPMP and the exact cycle sum problem are all polynomially equivalent.*

We briefly recall the ideas behind the reduction steps from the l-RPMP to the ECPMP to the exact cycle sum problem. Detailed descriptions of the reduction steps can be found in the proof of Theorem 4.4 in [46].

Proof (Outline). Concerning the first reduction step, Papadimitriou and Yannakakis use a generalized variant of the ECPMP in which all edges in the graph are weighted and a perfect matching with total weight equal to a given value is sought. This problem is called the *exact weight perfect matching problem* (EWPMP). The l-RPMP with upper bound constraints referring to the subsets $R_1, \ldots, R_l$ is polynomially reduced to the EWPMP with weights encoded in unary. This is done using edge weights which uniquely indicate in which of the subsets $R_1, \ldots, R_l$ an edge is contained. Then, there exists a solution matching to the l-RPMP if and only if there exists a solution matching among a series of EWPMP instances which differ in the demanded weights. The encoding lengths of the weights and the number of problem instances of the EWPMP both are polynomially bounded above, which is due to the fact that the parameter l of the the l-RPMP instance is fixed.

Next, EWPMP with weights encoded in unary is polynomially reduced to the ECPMP with a cardinality restriction referring to a subset of edges R. Each edge with weight w is replaced by a path having $2w - 1$ edges. In such a path exactly w node-disjoint edges are in R. Adding paths whose lengths linearly depend on weights is possible here since all weights are encoded in unary.

For the next step the authors use a generalized variant of the exact cycle sum problem, where all arcs in the digraph are weighted and the task is to find a set of node-disjoint cycles with total weight equal to a given value. We call this the *exact weight cycle sum problem.* Then, the ECPMP is polynomially reduced to the exact weight cycle sum problem with weights encoded in unary. Starting from an arbitrary perfect matching M in the graph of the problem instance of the ECPMP, the digraph in the problem instance of exact weight cycle sum is defined such that all possibilities to change the matching mate of a node u are represented by an arc emanating from u and ending at a new matching mate. The weight of such an arc corresponds to the difference in the total weight of the matching after and before changing the matching mate. Let R and k be the subset of edges and the cardinality parameter in the additional equality constraint in the instance of the ECPMP. In order to see if (and how) M can be changed into a solution matching of the ECPMP instance, the demanded total weight of all cycles is defined to be $k - |M \cap R|$.

At last, the exact weight cycle sum problem with weights encoded in unary is polynomially reduced to the exact cycle sum problem. The encoding of the weights in unary allows us to replace each arc with weight w by a path of w single arcs. $\square$

The problems where weights are encoded in unary in the outline of the proof of Theorem 4.4 can be seen as intermediate problems in the entire reduction sequence. Regarding the reduction from the l-RPMP to the EWPMP with weights encoded in unary, this reduction is polynomial due to the following: The values of the weights depend on the number of edges in the underlying graph and the parameter l. Since l is a constant, each weight can be written down in unary in polynomial time. When the EWPMP with weights encoded in unary is polynomially reduced to other problems, the encoding scheme of the weights allows reduction steps which are polynomial in the values of the weights, rather than in their logarithmic values.

One should note that although Papadimitriou and Yannakakis have shown the polynomial equivalence of the above problems, the question remains whether these problems are NP-hard or polynomially solvable. This remains an open question.

Nevertheless, we use their result to compare the complexity of the LCPMP to the complexity of these problems. We show that the LCPMP is of the same complexity as the ECPMP, the l-RPMP and the exact cycle sum problem. In Theorem 4.5 we prove the polynomial equivalence of the LCPMP to these three problems. In the corresponding proof, the LCPMP is polynomially reduced to ECPMP and the exact cycle sum problem is polynomially reduced to LCPMP.

Theorem 4.5. *For each $l \geq 2$ the LCPMP is polynomially equivalent to the ECPMP, the l-RPMP and the exact cycle sum problem.*

Proof. As the additional equality constraint in the ECPMP generalizes the level constraint, it is clear that the LCPMP polynomially reduces to the ECPMP. What remains to be proven is that one of the three problems in Theorem 4.4 polynomially reduces to the LCPMP. To do this, we reduce from the exact cycle sum problem.

Let (D, k) be a problem instance of exact cycle sum, where $D = (V, A)$ is a directed graph with nodes $V = \{v_1, \dots, v_n\}$ and k is an integer with $0 \leq k \leq n$. We polynomially reduce (D, k) to a problem instance of the LCPMP on the bipartite graph $G = (V \mathbin{\dot\cup} V', E)$. The set $V' = \{v'_1, \dots, v'_n\}$ consists of copies of the nodes in V. For each arc $(v_i, v_j) \in A$ the edge set E contains the edge $[v_i, v'_j]$. Further, E contains the on-level edges $[v_i, v'_i]$ for all $i = 1, \dots, n$. The parameter k' of the level constraint is defined as $k' := n - k$. An example of this reduction is given in Figure 4.1.

A matching in G is interpreted as follows: If an off-level edge $[v_i, v'_j]$ with $i \neq j$ is in M, then the arc (v_i, v_j) is in the solution of the exact cycle sum problem. If an on-level edge $[v_i, v'_i]$ is in M, then the node v_i is not contained in any cycle of the solution of the exact cycle sum problem.

Now, we show that there is a perfect matching in G which contains exactly k' on-level edges if and only if there is a set of node-disjoint cycles of total length k in D. If G contains a perfect matching which satisfies the level constraint, the arcs building the required cycles to solve the exact cycle sum instance are as described in the above interpretation of a matching in G. As the matching is perfect, all nodes $v_i \in V$ and $v'_i \in V'$ are covered by an edge of the matching. Hence, for each node v_i in the digraph D either none of its incident arcs are chosen to be in a cycle or exactly one ingoing and one outgoing arc are chosen to be in a cycle. This implies that the selected arcs build node-disjoint cycles. The level constraint ensures that exactly $k' = n - k$ on-level edges are in a matching. Therefore, exactly $n - k$ nodes in the graph D are not contained in any cycle, which is equivalent to exactly k nodes being contained in cycles. This corresponds to the demanded total length of the cycles.

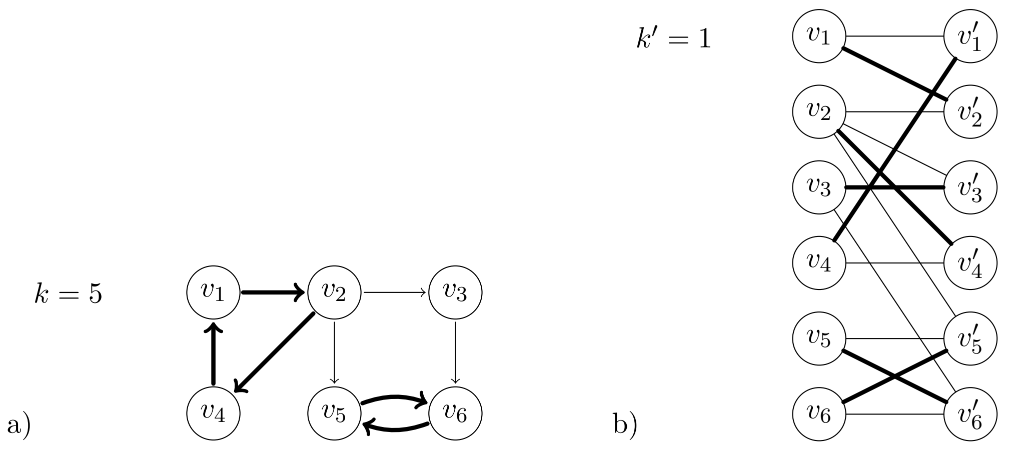


Figure 4.1.: a) Example of a problem instance of the exact cycle sum problem. A solution to it, which is a set of node-disjoint cycles of total length k, is indicated with bold arcs. b) Graph from problem reduction. A solution to the LCPMP, which is a perfect matching with exactly k' on-level edges, is indicated with bold edges.

Now, let there be a set of node-disjoint cycles of total length k in D. A solution matching of the LCPMP instance can be constructed by the following rule. For each arc (v_i, v_j) contained in one of the cycles, the off-level edge $[v_i, v'_j]$ is included in the matching. For each node v_i in the digraph D which is not contained in any cycle, the on-level edge $[v_i, v'_i]$ is included in the matching. As the cycles are node-disjoint, the resulting set of edges is a matching in G. The required total length of the cycles implies that exactly $n - k$ nodes in H are not contained in any cycle. Therefore, there are exactly k' on-level edges in the matching.

The problem reduction is polynomial, as $|V \cup V'| = 2n$ and $|E| = |A| + n$, and as the construction of a solution matching of the LCPMP out of a set of cycles can be done in $\mathcal{O}(|V|)$. □

One should note that in the reduction of the exact cycle sum problem to the LCPMP the graph of the LCPMP instance is a level graph. Hence, the statement of Theorem 4.5 stays valid for the LCPMP being restricted to level graphs.

Theorem 4.5 has interesting consequences. The polynomial equivalence of the LCPMP and the ECPMP proves that the level constraint of the LCPMP and the equality constraint of the ECPMP both have the same impact on the complexity of a perfect matching problem. This holds true even though the level constraint is a special case of an equality constraint. The differences in the two side constraints show up in the support of the corresponding equalities. While the support of an equality constraint in the ECPMP contains variables corresponding to the edges in any subset R of edges, the support of a level constraint in the LCPMP has a fixed structure. Each variable appearing in it corresponds to an on-level edge.

4.2. Assumptions decisive for the complexity of restricted matching

Under the assumption that P $\neq$ NP, the restricted perfect matching problem as defined in Section 4.1 is on the borderline between being polynomially solvable and being NP-hard. Its complexity depends on details of its problem formulation. Depending on whether certain problem parameters are fixed or not, the problem is polynomially solvable or NP-hard. We will show which changes in the formulation of the upper bound side constraints in restricted perfect matching problems are sufficient for the problem to be polynomially solvable or NP-hard. This problem is well qualified for such investigations, as we can modify not only the parameters within the upper bound side constraints, but also the number of these constraints. Further, there is a variant of a restricted perfect matching problem introduced by Itai et. al [29], which is NP-hard.

We begin with the restricted perfect matching problem as formulated and analyzed by Itai et. al [29] and discuss its main properties. As this problem variant is not restricted in the number of upper bound constraints, it can be seen as the general class of restricted perfect matching problems. Then, we recall the formulation of the l-RPMP as defined in Section 4.1. In this problem the number of upper bound side constraints is fixed. Two variants of restricted perfect matching problems which are both polynomially solvable will be presented afterwards. Finally, the problem variants are compared to each other concerning their complexity. A graphical listing of the different problem variants and their relation to each other can be found in Figure 4.2.

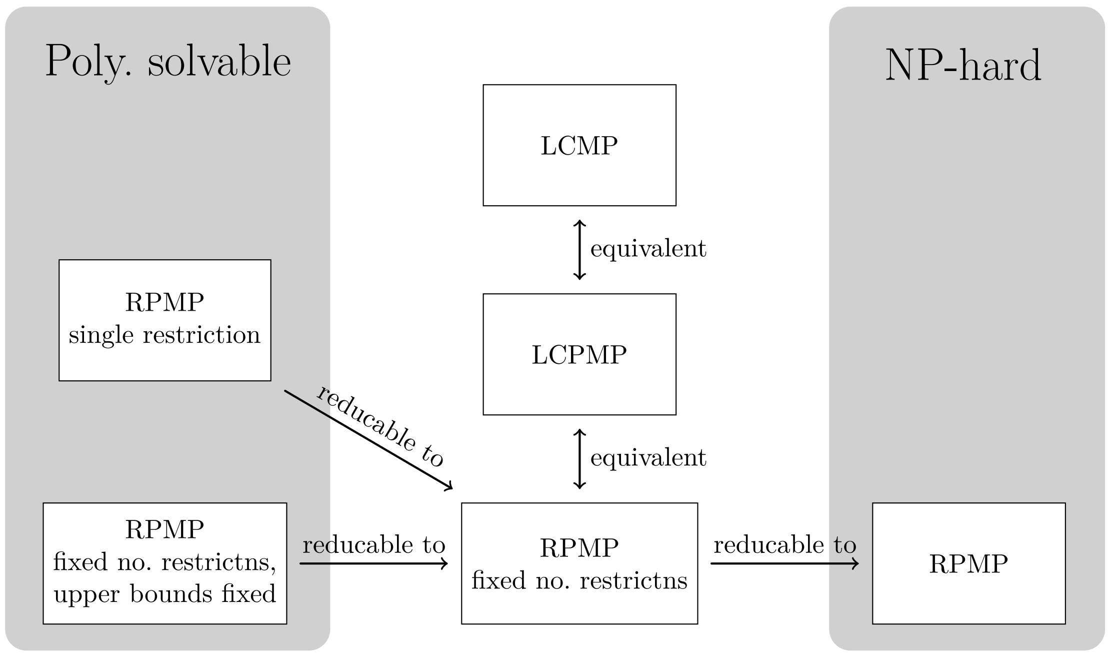


Figure 4.2.: Complexity relationships between the level constrained (perfect) matching problem and restricted perfect matching problems.

All problems considered in this section are stated on a bipartite graph $G = (U \mathbin{\dot\cup} V, E)$ with $U = \{u_1, \dots, u_n\}$ and $V = \{v_1, \dots, v_n\}$. In any problem variant of the restricted perfect matching problem, we refer to its upper bound side constraints as *restrictions.*

4.2.1. Restricted perfect matching with a variable number of restrictions

Itai et. al [29] define the general *restricted perfect matching problem* as follows:

Problem Formulation 4.6 (Restricted perfect matching)**.** *Let $R_1, \dots, R_k \subseteq E$ and let $r_1, \dots, r_k$ be positive integer values. The* restricted perfect matching problem (RPMP) *is the problem of finding a perfect matching M in G, which satisfies*

$$|M \cap R_i| \leq r_i \quad \textit{for all } i = 1, \dots, k.$$

The number of these restrictions, k, is part of the input and is not fixed.

One should mention that Itai et. al also present variants of this problem as complete and maximum matching problem.

In the RPMP, the number of upper bound side constraints is not fixed in advance, it rather is specified by the problem input. In other words, each problem instance of the RPMP may contain an arbitrary number of them. This fact is used by Itai et. al to prove that the RPMP is NP-hard (see [29]).

Itai et. al prove that the restricted complete matching problem is NP-hard, which directly results in the fact that the RPMP is NP-hard. The authors show a reduction of the satisfiability problem of boolean expressions in normal form (see Problem Formulation C.12 in Appendix C) to the restricted complete matching problem. Each clause is identified by exactly one node in the first color class of the underlying bipartite graph. In the second color class there is exactly one node for each occurrence of each literal in a clause. A node representing a clause and a node representing a literal of a clause are connected by an edge if and only if the two clauses coincide. An edge in a matching indicates that the literal corresponding to an end-node of the edge is chosen to be in the solution of the satisfiability problem. Hence, it must be prevented that the matching contains two edges with end-nodes standing for literals that are complements of each other. This is done with the help of the restrictions and the corresponding subsets of edges $R_1, \dots, R_k$. Each two edges with end-nodes standing for literals that are complements of each other build a set R_i. The upper bound on the number of matching edges in a set R_i is set to 1 for all $i = 1 \dots, k$. At this point it shows that the number of restrictions cannot be fixed, as with an increasing number of clause nodes and literal nodes the number of sets R_i also increases. This prevents the parameter k from being fixed.

4.2.2. Restricted perfect matching with a fixed number of restrictions

As introduced in Section 4.1, the l-RPMP is a restricted perfect matching problem with a *fixed* number of restrictions

$$|M \cap R_i| \leq r_i \quad \text{for all } i = 1, \dots, l.$$

It is stated in Theorem 4.4 that for $l \geq 2$ the l-RPMP is polynomially equivalent to the ECPMP. As mentioned before, the complexity of the ECPMP and hence, the complexity of the l-RPMP for each $l \geq 2$, is unknown. On the other hand, we stated in Theorem 4.5 that for each $l \geq 2$ the l-RPMP is polynomially equivalent to the LCPMP, as well, and for the LCPMP we develop an approximation algorithm in Section 7.1.

A crucial condition in the problem formulation of the l-RPMP is that the number of restrictions l (and thus, the number of sets $R_1, \dots, R_l$) is fixed. This is exactly where this problem and the problem variant by Itai et. al differ from each other.

4.2.3. Restricted perfect matching with a single restriction

For the special case when there is only one restriction in the restricted perfect matching problem, there exists a polynomial solution to it.

Problem Formulation 4.7 (Restricted perfect matching with single restriction)**.** *The* restricted perfect matching problem with a single restriction (1-RPMP) *corresponds to the l-RPMP with $l = 1$.*

Itai et. al [29] offer a solution strategy based on the reformulation of the problem as a minimum cost flow problem. We present their result in the next theorem:

Theorem 4.8. *The restricted perfect matching problem with a single restriction is polynomially solvable.*

Proof. Let $(G = (U \mathbin{\dot\cup} V, E), R_1, r_1)$ be a problem instance of the 1-RPMP, where G is a bipartite graph with $|U| = |V| = n$, the set R_1 is a subset of E and r_1 is a positive integer. The problem instance of the resulting minimum cost flow problem is denoted by $(D = (N, A), s, t, p, c, b)$, with distinct source and sink nodes s and t in N, arc capacities p_a for all $a \in A$, arc costs c_a for all $a \in A$ and supply/demand values b_z for all $z \in N$. The digraph D and the parameters of the minimum cost flow problem are defined as follows:

$$\begin{aligned}
&N = U \cup V \cup \{s, t\}, && \text{where } s, t \notin U \cup V,\\
&A = \{(s, u) \mid u \in U\} \cup \{(v, t) \mid v \in V\} &&\\
&\qquad \cup \{(u, v) \mid [u, v] \in E, u \in U, v \in V\}, &&\\
&p_a = 1, && \text{for all } a \in A,\\
&c_{(u,v)} = \begin{cases} 1 & \text{if } [u, v] \in R_1,\\ 0 & \text{else,} \end{cases} &&\\
&b_z = \begin{cases} n & \text{if } z = s,\\ 0 & \text{for all } z \in N \setminus \{s, t\},\\ -n & \text{if } z = t. \end{cases} &&
\end{aligned}$$

We can assume that there is a flow in D which has a flow value of n, as otherwise the graph G would not contain any perfect matching. The digraph D is created following the usual approach when reformulating a minimum cost perfect matching problem on a bipartite graph as a minimum cost flow problem (compare Section 1.3.2).

The fact that there is only one restriction in the 1-RPMP allows us to “encode” this upper bound side constraint in terms of the costs of the arcs. All arcs that correspond to edges in R_1 are assigned a cost value of 1, all other arcs have cost 0. Therefore, the total cost of a flow in D equals the number of edges in the matching which are in R_1. Since the aim is to find a feasible flow of minimum cost, there exists a solution of total cost less than or equal to r_1 if and only if there is a perfect matching M in G which satisfies $|M \cap R_1| \leq r_1$. The minimum cost flow problem can be solved in polynomial time (see Section 1.1.2), e.g. by the successive shortest path method, therefore, the 1-RPMP can be solved in polynomial time. □

It is interesting to see that reformulating an instance of the RPMP as a minimum cost flow problem does not work this way when there are two or more restrictions. In the case of a single restriction, a solution to the minimum cost flow problem has a minimum number of arcs corresponding to edges in R_1. So, the restriction is incorporated into the objective function of the minimum cost flow problem. The value of the upper bound r_1 does not appear in its formulation. In the case of two restrictions, there is not a single set of arcs anymore out of which a minimum number of arcs are chosen. There are rather two sets of arcs which need to be considered separately, both with their own upper bound value. This cannot be modeled by a single objective function. So, the RPMP with two upper restrictions belongs to the class of resource constrained problems whose complexity is unknown.

The situation is different when additionally to a fixed number of restrictions in the l-RCMP we demand that all upper bounds $r_1, \dots, r_l$ are bounded above by a fixed value.

4.2.4. Restricted perfect matching with a fixed number of restrictions and a fixed bound on the restrictions’ upper bound values

Let l and r be two fixed, positive integers. We now consider the l-RPMP on the graph G with restrictions

$$|M \cap R_i| \leq r_i, \quad \text{for all } i = 1, \dots, l,$$

with $r_i \leq r$ for all $i = 1, \dots, l$.

In contrast to the l-RPMP, in this problem each upper bound value of the restrictions is bounded above by the fixed value r. We show that this variant of the RPMP is polynomially solvable.

Theorem 4.9. *Let $G = (V, E)$ be a graph and let $m := |E|$. Further, let l and r be two fixed, positive integers.*

The l-RPMP with all upper bound values of the restrictions bounded above by r can be solved on G in $\mathcal{O}(m^{rl} \cdot \mathrm{PM}(G))$, where $\mathrm{PM}(G)$ is the time needed to solve a perfect matching problem on the graph G.

Proof. Let $m_i := |R_i|$ for all $i = 1, \ldots, l$. The number of possibilities to choose at most r_i edges from the set R_i equals $\sum_{j=0}^{r_i} \binom{m_i}{j}$ for all $i = 1, \ldots, l$. For the number of combinations of all possibilities over all sets $R_1, \ldots, R_l$ it holds that

$$\prod_{i=1}^{l} \sum_{j=0}^{r_i} \binom{m_i}{j} \leq \prod_{i=1}^{l} \sum_{j=0}^{r} \binom{m}{j} = \prod_{i=1}^{l} \sum_{j=0}^{r} \frac{m!}{(m-j)!j!} \leq \prod_{i=1}^{l} m^r = m^{rl}. \tag{4.3}$$

One should mention that in order to determine the exact number of combinations, all edges appearing in more than one set R_i need to be treated with special care. As these edges decrease the number of possible combinations, estimation (4.3) stays valid. So, the number of possible combinations to choose edges out of $R_1, \ldots, R_l$ such that the restrictions are satisfied, is in $\mathcal{O}(m^{rl})$. As l and r are fixed, the number of possible combinations is asymptotically bounded by a polynomial in m.

In order to find out whether there is a perfect matching in G which satisfies the restrictions, we proceed similarly to what we presented in Section 2.3. For each combination of edges as described above, we first check if it fulfills the matching conditions. If it does not, then the corresponding combination is discarded. If it does, we remove all edges in the current combination from G including their end-nodes. Further, all edges in $\bigcup_{i=1}^{l} R_i$ which are not in the current combination are also removed from G. Let $\bar{G}$ denote the resulting graph. Then, we search for a perfect matching in $\bar{G}$. This can be done in polynomial time (see Section 1.3.4). If there is a perfect matching in $\bar{G}$, then this matching can be extended by the edges of the current combination to a perfect matching in G which satisfies the upper bound restrictions.

Hence, the l-RPMP with a fixed bound on the upper bound values of the restrictions can be solved polynomially by enumerating all feasible combinations of edges out of $R_1, \ldots, R_l$ and using a polynomial-time perfect matching algorithm. The total running-time of this procedure is $\mathcal{O}(m^{rl} \cdot \mathrm{PM}(G))$, where $\mathrm{PM}(G)$ is the time needed to solve a perfect matching problem in the graph G, and l and r are the fixed values from the problem formulation. □

4.2.5. Comparison of the restricted perfect matching problem variants

When comparing the different problem variants of the restricted perfect matching problems described in this section, we find that the complexity of the problem depends on whether the number of restrictions, i.e. the number of additional upper bound side constraints, is fixed or not. More specifically, when the number of side constraints is not fixed, then the RPMP is NP-hard. This fact is independent from the upper bound values of the restrictions. As one can see in the corresponding NP-hardness proof of Itai et. al [29], all upper bounds used are of value 1. So, the RPMP with a variable number of restrictions is NP-hard even when the upper bounds are fixed to 1.

On the other hand, when the number of restrictions is fixed, it then holds that the problem can be solved polynomially if the upper bound values in the restrictions are bounded above by a fixed value.

Situated "between" these two problem variants is the l-RPMP where the number of restrictions is fixed but the upper bound values are not bounded above by a fixed value.

Although its classification as polynomially solvable or NP-hard is still open, we presented modifications in its problem formulation which are sufficient for the problem to become polynomially solvable or NP-hard.

The problem variant with a single restriction has a special position in the classification of the problems. We have seen that it is polynomially solvable due to the fact that there is exactly one restriction, independent from the value of its upper bound. As a summary of this section we refer to Figure 4.2 again, where the complexity relationship between all problem variants discussed in this section and their relationship to the LCPMP (and the LCMP) is depicted.

4.3. The complexity of assignment problems with an additional equality side constraint

Next, we consider assignment problems which have imposed an additional equality constraint. In Section 2.2.3 we defined the level constrained assignment problem, which corresponds to the special case where the additional equality constraint is applied to the set of on-level edges. In order to be able to name single assignments explicitly, we follow the notion of formulating the problem on a complete bipartite graph.

4.3.1. Equality constraint on all edges of the graph

We start with a variant of an additional equality side constraint for which the problem turns out to be polynomially solvable. The problem is an assignment problem where the size of the assignment needs to equal a given parameter k. Dell'Amico and Martello refer to this problem as the *k-cardinality assignment problem* (k-AP) (see [18]). We call this problem the *fixed size assignment problem*, in order to avoid the parameter k appearing in the problem name, as k is part of the problem input.

Formally, let $m, n \in \mathbb{N}$ and let k be a nonnegative integer with $k \leq \min(m, n)$. Let $c_{ij} \in \mathbb{R}$ be the cost of the edge $[i, j]$ for all $i = 1, \ldots, m$ and $j = 1, \ldots, n$.

Problem Formulation 4.10 (Fixed size assignment)**.** *The* fixed size assignment problem (FSAP) *is stated as the integer linear program*

$$\begin{aligned}
\min \quad & \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} && && \\
s.t. \quad & \sum_{j=1}^{n} x_{ij} \leq 1 && \forall\, i = 1, \ldots, m && (4.4)\\
& \sum_{i=1}^{m} x_{ij} \leq 1 && \forall\, j = 1, \ldots, n && (4.5)\\
& \sum_{i=1}^{m} \sum_{j=1}^{n} x_{ij} = k && && (4.6)\\
& x_{ij} \in \{0, 1\} && \forall\, i = 1, \ldots, m, \forall\, j = 1, \ldots, n. &&
\end{aligned}$$

A noticeable difference to the formulation of assignment problems in Section 1.2.3 and Section 2.2.3 lies in the constraints (4.4), (4.5). In the formulation of the classical assignment problem these constraints are equalities. Here, these constraints are inequalities.

One should note that the parameter k is part of the input of the problem. Otherwise, the problem could be solved easily as follows. As the assignment is supposed to contain exactly k of the mn edges, we can find an optimal solution by checking all $\binom{mn}{k}$ possible solutions for the best objective value. We have that

$$\binom{mn}{k} = \frac{(mn)!}{(mn-k)!k!} \in \mathcal{O}\left((mn)^k\right).$$

Hence, if k were fixed, then this would yield a polynomial solution procedure.

Dell'Amico and Martello [18] show that the FSAP with k not being fixed is polynomially solvable. They offer three different polynomial solution approaches.

Firstly, they interpret the FSAP as a problem on an intersection of two matroids M_1 and M_2. In order to define the matroids, let the underlying graph of the FSAP be $K_{m,n} = (U \uplus V, E)$. Then, $M_1 = (E, \mathcal{I}_1)$ is the matroid on the ground set E, where $\mathcal{I}_1$ is the set of all subsets $I \subseteq E$ with $|I| \leq k$ and with no two edges in I sharing an end-node in U. The matroid $M_2 = (E, \mathcal{I}_2)$ has the same ground set and $\mathcal{I}_2$ is the set of all subsets $I \subseteq E$ with $|I| \leq k$ and with no two edges in I sharing an end-node in V. Now, the FSAP corresponds to the problem of finding a minimum cost subset of E which is contained in both sets $\mathcal{I}_1$ and $\mathcal{I}_2$, and which is of cardinality k. Each basis of M_1 or M_2 is an independent set of maximum cardinality in the corresponding matroid. The FSAP is feasible if and only if there exists a common basis B of the two matroids with $|B| = k$. In order to find a common basis, the weighted matroid intersection algorithm (see Lawler [37], Edmonds [21]) can be used. This algorithm finds a common basis of M_1 and M_2 which is of minimum cost in polynomial time. In order to solve the FSAP, we determine this basis and check whether it is of cardinality k or not.

The second solution approach to the FSAP is to show that the constraint matrix of the problem is totally unimodular. Let $B \in \{0,1\}^{(m+n)\times mn}$ be the constraint matrix corresponding to the inequalities (4.4) and (4.5). By adding $m+n$ slack variables, the inequalities can be transformed into equalities, i.e. the problem is in standard form. The complete constraint matrix of the FSAP in standard form is the $(m+n+1)\times(mn+m+n)$ matrix

$$A = \left(\begin{array}{c|c} B & I \\ \hline 1\dots1 & 0\dots0 \end{array}\right),$$

where I is the identity matrix I_{m+n}. Dell'Amico and Martello prove that the matrix A is totally unimodular by using the well-known partitioning argument (see [43]). Hence, the FSAP can be solved polynomially using linear programming techniques.

The third alternative to solve the FSAP polynomially is by modeling the problem as a minimum cost flow problem. The digraph for this problem is constructed as usual when transforming a weighted bipartite matching problem into a minimum cost flow problem (see Section 1.3.2). Then, the FSAP corresponds to the problem of finding a b-flow of value k which is of minimum cost.

For the case of a sparse cost matrix $(c_{ij})_{ij}$, Dell'Amico et. al [17] derive an efficient algorithm which is faster than the minimum cost flow approach.

All solution approaches from above have in common that they make use of the fact that the equality constraint (4.6) contains *all* variables with non-zero coefficients. In other words, it refers to *all* of the edges in the graph. In contrast to this constraint, the level constraint applies its cardinality restriction only to a subset of the edges – the on-level edges.

Unfortunately, the solution approaches for the FSAP cannot be applied in the case where the additional cardinality constraint refers to a general subset R of the edges. Nevertheless, the problem of finding an assignment which satisfies an additional equality constraint on a general subset of the edges can be solved polynomially. The next section deals with this problem.

4.3.2. Equality constraint on a subset of edges of the graph

Now, we generalize the set of edges which are affected by the additional equality constraint. We consider a perfect matching problem on the complete bipartite graph $K_{n,n} = (U \cup V, E)$ with an additional cardinality constraint demanding an exact number of edges of a given set $R \subseteq E$ to be in the matching. Let k denote that number, with k being an integer-valued parameter with $0 \leq k \leq n$.

Problem Formulation 4.11 (Equality constrained assignment)**.** *The* equality constrained assignment problem (ECAP) *is the problem of finding a solution to the system*

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad \forall\ i = 1, \dots, n \tag{4.7}$$

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad \forall\ j = 1, \dots, n \tag{4.8}$$

$$\sum_{[u_i, v_j] \in R} x_{ij} = k \tag{4.9}$$

$$x_{ij} \in \{0, 1\} \qquad \forall\ i, j = 1, \dots, n. \tag{4.10}$$

In order to coincide with the formulation of this problem in the literature, the ECAP is stated as a feasibility problem without objective function. In the case that the set R is of the form $R = \{[u_i, v_i] \mid i = 1, \dots, n\}$, the constraint (4.9) becomes the level constraint and the ECAP coincides with the LCAP from Section 2.2 with all weights being equal to 1.

Karzanov [32] was the first who gave necessary and sufficient conditions for the existence of a feasible solution to the ECAP. Yi, Murty and Spera [52] simplified the proofs of these conditions and used them to develop a polynomial algorithm to solve the ECAP:

Theorem 4.12. *The equality constrained assignment problem on the complete bipartite graph $K_{n,n}$ can be solved in $\mathcal{O}(n^{2.5})$.*

The authors derive the algorithm and the corresponding conditions in two steps. First, they restrict the graph G to not contain any so-called odd 2×2 subgraphs. These are subgraphs of G induced by two nodes of each color class, U and V, which have an odd number of edges in R. Next, they consider the case when $G[R]$ contains a matching M_R of size k and $G[E \setminus R]$ contains a matching $M_{\bar{R}}$ of size $n-k$. The two matchings together become, after some modifications, a solution matching to the ECAP. The modifications steps are made to avoid that nodes are covered by two edges in M_R and $M_{\bar{R}}$. A detailed description of the algorithm can be found in [52].

In the formulation of the ECAP ((4.7) – (4.10)), no objective function is considered. Adding the objective function of the classical assignment problem to it yields the corresponding optimization problem

$$\begin{aligned} &\min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \\ &\text{s.t. } (4.7) - (4.10). \end{aligned}$$

No polynomial time algorithm for solving this problem is known yet. This fact also is of importance for the LCMP, as the LCMP is a special case of this problem. This can be seen when setting R to be the set of on-level edges and setting $c_{ij} := -1$ for all edges $[u_i, v_j]$ that are contained in the graph of the LCMP instance, and $c_{ij} := 0$ otherwise. Then, the above problem has an optimal solution of total cost $-z$ if and only if the LCMP instance has an optimal solution of size z. Even though there is no polynomial time algorithm for the optimization variant of the ECAP yet, Alfakih et. al [7] present classes of facets of the ECAP polytope for the special case which they refer to as the partitioned case (see Section 6.2).

Let us consider the original formulation of the ECAP as a feasibility problem again. Now, we go one step further in generalizing the equality constraint (4.9) and replace it with

$$\sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} = k \tag{4.11}$$

When c_{ij} is interpreted as the cost of the edge $[u_i, v_j]$, constraint (4.11) requires that the total cost of a perfect matching is equal to k. If $c_{ij} \in \{0, 1\}$ for all $i, j = 1, \dots, n$, constraint (4.11) corresponds to constraint (4.9) with R being defined appropriately.

For c_{ij} being general nonnegative integers for all $i, j = 1, \dots, n$, this problem is NP-complete. Chandrasekaran et. al [13] prove this by showing that the subset sum problem is a special case of this problem.

5. The Complexity of Couple and Level Constrained Matching

In this chapter, we investigate the relationship between couple constraints and the level constraint, and we determine the complexity of a new resource constrained matching problem where both types of side constraints appear together.

In Section 5.1 we show that the level constrained matching problem can be polynomially reduced to the couple constrained matching problem. In Section 5.2 we prove that the decision version of the couple and level constrained matching problem is NP-complete. In this problem, the couples are restricted to contain only on-level edges. It holds that if the level constraint is removed, the resulting couple constrained matching problem instances are trivial to solve. Thus, our complexity result also shows that adding a single level constraint to a polynomially solvable class of couple constrained matching problems suffices to induce an NP-hard problem.

5.1. Couple constraints as generalization of the level constraint

Couple constraints, together with a simple modification of the underlying graph, can be used to model a level constraint. In other words, the couple constraints generalize the level constraint. We show this by giving a polynomial reduction of the LCPMP to the CCPMP.

Theorem 5.1. *The level constrained perfect matching problem can be polynomially reduced to the couple constrained perfect matching problem.*

Proof. We start with a problem instance $\big(G, k\big)$ of the LCPMP, where $G = (U \mathbin{\dot\cup} V, E)$ is a bipartite graph with $U = \{u_1, \ldots, u_n\}, V = \{v_1, \ldots, v_n\}$, and k is an integer with $0 \leq k \leq n$. Without loss of generality, let $[u_1, v_1], \ldots, [u_p, v_p]$ be the on-level edges occurring in G. Based on G and k, we build a problem instance of the CCPMP. It consists of a graph $G' = \big((U \cup U') \mathbin{\dot\cup} (V \cup V'), E \cup E'\big)$ and a couple collection $\mathcal{F} = \{F_1, \ldots, F_p\}$. Figure 5.1 illustrates the construction of the graph G' and the couple collection $\mathcal{F}$.

The sets U' and V' are defined as follows:

$$U' := \{u'_i \mid i = 1, \ldots, p\} \cup \{x_i \mid i = 1, \ldots, p - k\},$$
$$V' := \{v'_i \mid i = 1, \ldots, p\} \cup \{y_i \mid i = 1, \ldots, p - k\}.$$

The set E' contains the following edges between U' and V':

$$\begin{aligned} E' := &\{[u'_i, v'_i] \mid i = 1, \ldots, p\} \\ &\cup \{[u'_i, y_j] \mid i = 1, \ldots, p; j = 1, \ldots, p-k\} \\ &\cup \{[v'_j, x_j] \mid i = 1, \ldots, p; j = 1, \ldots, p-k\}. \end{aligned}$$

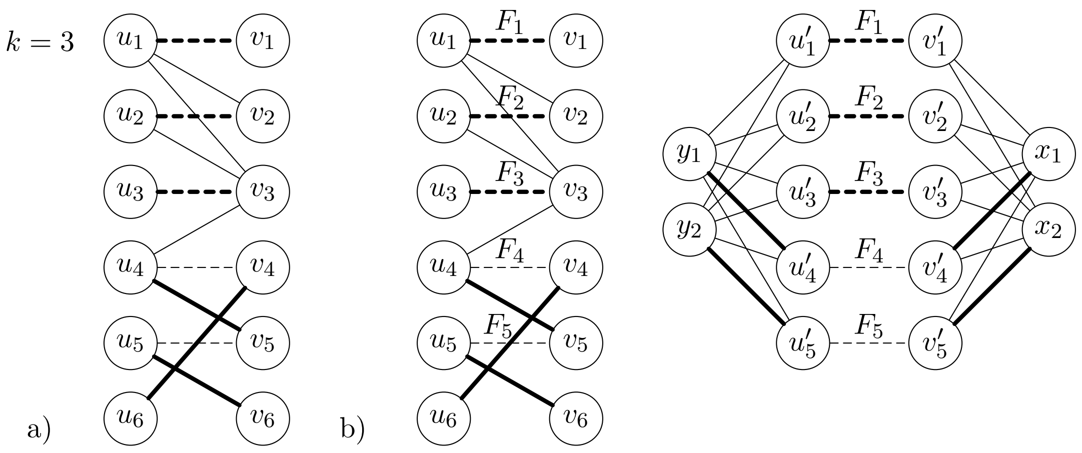
The couples of the couple collection $\mathcal{F}$ are defined to be the following:

$$F_i := \{[u_i, v_i], [u'_i, v'_i]\}, \quad \text{for } i = 1, \ldots, p.$$

The couple constraints ensure that an on-level edge $[u_i, v_i]$ in E is chosen to be in the solution matching if and only if the corresponding on-level edge $[u'_i, v'_i]$ in E' is chosen to be in it.

Now, we consider a perfect matching in G'. One can see that each perfect matching in G' contains exactly k on-level edges of the form $[u'_i, v'_i]$: If there would be less than k, then not all of the nodes $u'_i, i = 1, \ldots, p$, could be covered by the matching. Otherwise, if there would be more than k, then not all of the nodes $x_j, j = 1, \ldots, p-k$, could be covered by the matching. This means that the entire graph G' contains a perfect matching satisfying the couple constraints regarding $\mathcal{F}$ if and only if the subgraph G has a perfect matching containing exactly k on-level edges.

The construction of the problem instance of the CCPMP is polynomial in the input size of the problem instance of the LCPMP, as $|U'|, |V'| \leq 2n$ and $|\mathcal{F}| = p \leq n$. This completes the proof. $\square$



$k = 3$
u_1 u_2 u_3 u_4 u_5 u_6
v_1 v_2 v_3 v_4 v_5 v_6
a)
b)
F_1 F_2 F_3 F_4 F_5
u'_1 u'_2 u'_3 u'_4 u'_5
v'_1 v'_2 v'_3 v'_4 v'_5
y_1 y_2
x_1 x_2

Figure 5.1.: a) Example of an LCPMP instance with $k = 3$. All on-level edges are dashed. A perfect matching which contains exactly k on-level edges is indicated with bold edges. b) CCPMP instance resulting from problem reduction. The couples to which the on-level edges belong are noted above the on-level edges. Bold edges indicate a perfect matching which satisfies the couple constraints.

5.2. The complexity of the couple and level constrained matching problem

In this section, we investigate the case where couple constraints and a level constraint appear together as additional side constraints in a matching problem. In formulating the problem, these two types of side constraints get imposed on a matching problem one after the other. We start with a polynomially solvable special case of the CCMP. We will then prove that it suffices to add a single level constraint to it, in order to make the resulting problem – the couple and level constrained matching problem – NP-hard. This problem has not been considered in the literature before.

5.2.1. The couple constrained matching problem with on-level couples

The starting point of our considerations is a class of couple constrained matching problems which can be solved in polynomial time. We have shown in Section 3.2 that the CCMP is NP-hard in general. At the end of Section 3.3, we have already mentioned a special case of the CCMP which is polynomially solvable (see Lemma 3.16). There, we considered the CCMP stated on a complete bipartite graph $K_{n,n}$ with all couples containing on-level edges only. This special case of the CCMP is polynomially solvable. An optimal solution can be found easily by choosing all on-level edges to be in a matching. Obviously, the restriction on the couples to contain only on-level edges makes this an optimal solution to the CCMP on all level graphs. According to these observations, we now define the class of CCMP instances which fulfill these restrictions on couples.

Let $G = (U \cupdot V, E)$ be a level graph, with $U = \{u_1, \ldots, u_n\}$ and $V = \{v_1, \ldots, v_n\}$. Further, let $\mathcal{F} = \{F_1, \ldots, F_p\}$ be a couple collection with $F_r = \{[u_{2r-1}, v_{2r-1}], [u_{2r}, v_{2r}]\}$ for all $r = 1, \ldots, p$.

Problem Formulation 5.2 (Couple constrained matching with on-level couples)**.** *The* couple constrained matching problem with on-level couples (CCMP-L) *is formulated as the integer linear program*

$$
\begin{array}{lll}
\max & \sum\limits_{e \in E} x_e & \\
s.t. & \sum\limits_{e \in \delta(u_i)} x_e \le 1 & \forall\ i = 1, \ldots, n \\
& \sum\limits_{e \in \delta(v_i)} x_e \le 1 & \forall\ i = 1, \ldots, n \\
& x_{2r-1,2r-1} = x_{2r,2r} & \forall\ r = 1, \ldots, p \\
& x_e \in \{0, 1\} & \forall\ e \in E.
\end{array}
$$

It is important to note that the graphs in problem instances of the CCMP-L are level graphs. As we have mentioned above, for each problem instance of the CCMP-L which contains a level graph $G = (U \cup V, E)$ with $2n$ nodes, an optimal solution x^* is given by

$$\begin{aligned} x^*_{ii} &= 1 \quad \text{for all } i = 1, \dots, n, \\ x^*_{ij} &= 0 \quad \text{for all } [u_i, v_j] \in E \text{ with } i \neq j. \end{aligned}$$

Lemma 5.3. *The couple constrained matching problem with on-level couples is polynomially solvable.*

The fact that all couples in an instance of the CCMP-L consist of on-level edges only, is a necessary condition for the problem to be polynomially solvable. If this restriction on the couples is left out, we get the (original) CCMP on level graphs, which is NP-hard, as we have shown in Corollary 3.12.

One should note that the reason for the CCMP-L being polynomially solvable does not lie in fixing the number of couple constraints. The number of couple constraints in the CCMP-L can grow linearly with the number of nodes, as in any level graph with $2n$ nodes there can be up to $\frac{1}{2}n$ couples.

5.2.2. The couple and level constrained matching problem with on-level couples

Now, we will investigate what impact an additional resource constraint has on the easy-to-solve CCMP-L. To this end we impose a level constraint to the CCMP-L. The resulting problem is defined as follows.

Let $G = (U \cup V, E)$ be a level graph, with $U = \{u_1, \dots, u_n\}, V = \{v_1, \dots, v_n\}$ and let $\mathcal{F} = \{F_1, \dots, F_p\}$ be a couple collection with $F_r = \{[u_{2r-1}, v_{2r-1}], [u_{2r}, v_{2r}]\}$ for all $r = 1, \dots, p$. Further, let k be a nonnegative integer with $k \leq n$.

Problem Formulation 5.4 (Couple and level constrained matching with on-level couples)**.** *The* couple and level constrained matching problem with on-level couples (CLCMP-L) *is formulated as the integer linear program*

$$\begin{aligned} \max \quad & \sum_{e \in E} x_e && && (5.1) \\ s.t. \quad & \sum_{e \in \delta(u_i)} x_e \leq 1 && \forall\, i = 1, \dots, n && (5.2) \\ & \sum_{e \in \delta(v_i)} x_e \leq 1 && \forall\, i = 1, \dots, n && (5.3) \\ & x_{2r-1,2r-1} = x_{2r,2r} && \forall\, r = 1, \dots, p && (5.4) \\ & \sum_{i=1}^{n} x_{i,i} = k && && (5.5) \\ & x_e \in \{0, 1\} && \forall\, e \in E. && (5.6) \end{aligned}$$

We will prove that the decision version of the CLCMP-L is NP-complete. The decision version of the CLCMP-L is defined in the obvious way. In addition to $G, \mathcal{F}$ and k from above, an instance contains an integer l with $0 \leq l \leq n$. The CLCMP-L decision problem asks whether there is a matching M of size $|M| \geq l$ in G, which satisfies the couple constraints for $\mathcal{F}$ and the level constraint for k.

Building blocks

To prove NP-completeness of the CLCMP-L decision problem, we will polynomially reduce an arbitrary instance of the NP-complete clique decision problem to a problem instance of the CLCMP-L decision problem. In this instance, the graph and parts of the couple collection will be constructed modularly out of so-called *building blocks*. The building blocks are based on the parameters of the clique problem. We now present the construction of such a building block.

Consider a graph $H = (W_H, E_H)$ with $W_H = \{w_1, \ldots, w_n\}$ and an integer l with $0 \leq l \leq n$, which together constitute an instance of the clique decision problem. Each node $w \in W_H$ induces a building block, which consists of an auxiliary graph $G_w = (U_w \uplus V_w, E_w)$ and a couple collection $\mathcal{F}^b_w$. The construction of this building block depends on the neighbors of the node w and the parameter l. Let $\Gamma_H(w)$ denote the set of neighbors of the node w in the graph H. The auxiliary graph G_w consists of the two connected components, B_w and D_w, as shown in Figure 5.2.

The first subgraph $B_w = (V^B_w, E^B_w)$ is a bipartite graph whose node set V^B_w consists of the two color classes X_w and X'_w. They are defined as

$$\begin{aligned} X_w &:= \{a_{w,\tilde{w}} \mid \tilde{w} \in \Gamma_H(w)\} \cup \{b_{w,h} \mid h = 1, \ldots, l-1\}, \\ X'_w &:= \{a'_{w,\tilde{w}} \mid \tilde{w} \in \Gamma_H(w)\} \cup \{b'_{w,h} \mid h = 1, \ldots, l-1\}. \end{aligned}$$

So, for each edge that connects the node w with one of its neighbors, say $\tilde{w}$, in H, there is a node $a_{w,\tilde{w}}$ in X_w representing this adjacency. Further, X_w contains $l-1$ additional nodes $b_{w,h}$. The definition of the second color class X'_w is analogous to the definition of X_w.

The edge set E^B_w corresponding to the subgraph B_w contains two types of edges. First, it contains all on-level edges between X_w and X'_w. Second, it contains all edges connecting the nodes $a_{w,\tilde{w}}$ and $b'_{w,h}$ for all $a_{w,\tilde{w}} \in X_w$ and $b'_{w,h} \in X'_w$. Formally, the edge set E^B_w is defined as

$$\begin{aligned} E^B_w := &\left\{[a_{w,\tilde{w}}, a'_{w,\tilde{w}}], [b_{w,h}, b'_{w,h}] \mid \tilde{w} \in \Gamma_H(w); h = 1, \ldots, l-1\right\} \\ &\cup \left\{[a_{w,\tilde{w}}, b'_{w,h}] \mid \tilde{w} \in \Gamma_H(w); h = 1, \ldots, l-1\right\}. \end{aligned}$$

Besides the subgraph B_w, there is a second subgraph $D_w = (V^D_w, E^D_w)$. This subgraph is a bipartite level graph in form of a simple cycle. Its node set V^D_w consists of the two color classes Y_w and Y'_w. The subgraph D_w is defined as

$$\begin{aligned} Y_w &:= \{d_{w,h} \mid h = 1, \ldots, l-1\}, \\ Y'_w &:= \{d'_{w,h} \mid h = 1, \ldots, l-1\}, \\ E^D_w &:= \left\{[d_{w,h}, d'_{w,h}] \mid h = 1, \ldots, l-1\right\} \cup \left\{[d_{w,h}, d'_{w,h+1}] \mid h = 1, \ldots, l-1\right\}, \end{aligned}$$

with $d'_{w,l}$ being identified with the node $d'_{w,1}$, in order to close the cycle.

Next, we define the couple collection $\mathcal{F}^b_w = \{F^b_{w,h} \mid h = 1, \ldots, l-1\}$ by

$$F^b_{w,h} := \Big\{[b_{w,h}, b'_{w,h}], [d_{w,h}, d'_{w,h}]\Big\} \quad \text{for all } h = 1, \ldots, l-1.$$

The couples in $\mathcal{F}^b_w$ can be seen as links between the subgraphs B_w and D_w. In Figure 5.2, two edges of the same couple are marked with the same color.

The properties of the auxiliary graph G_w and the couple collection $\mathcal{F}^b_w$ are summarized in the following lemma.

Lemma 5.5. *Let $H = (W_H, E_H)$ be a graph with $W_H = \{w_1, \ldots, w_n\}$ and let l be an integer with $0 \le l \le n$. Furthermore, let w be an arbitrary node in W_H and let $\Gamma_H(w)$ be the set of neighbors of w. The graph G_w and the couple collection $\mathcal{F}^b_w$ are constructed as described above. Then, the following properties hold:*

- *The graph G_w is a bipartite level graph.*
- *The subgraphs B_w and D_w are node-disjoint. Each of them are connected components.*
- *The subgraph $B_w = (V^B_w, E^B_w)$ is of the following size:*
  - $|V^B_w| = 2\,(\deg_H(w) + l - 1)$,
  - *E^B_w contains $\deg_H(w) + l - 1$ on-level edges and $\deg_H(w)(l-1)$ off-level edges.*
- *The subgraph $D_w = (V^D_w, E^D_w)$ is of the following size:*
  - $|V^D_w| = 2(l-1)$,
  - *E^D_w contains $l-1$ on-level edges and $l-1$ off-level edges.*
- *The entire auxiliary graph $G_w = (U_w \uplus V_w, E_w)$ is of the following size:*
  - $|U_w| = |V_w| = \deg_H(w) + 2(l-1)$,
  - $|E_w| = \deg_H(w)l + 3(l-1)$.
- *The couple collection $\mathcal{F}^b_w$ consists of $l-1$ couples. All couples contain on-level edges only.*

NP-completeness of the couple and level constrained matching problem with on-level couples

We use the introduced building blocks to prove the main result in this section:

Theorem 5.6. *The decision version of the couple and level constrained matching problem with on-level couples is NP-complete.*

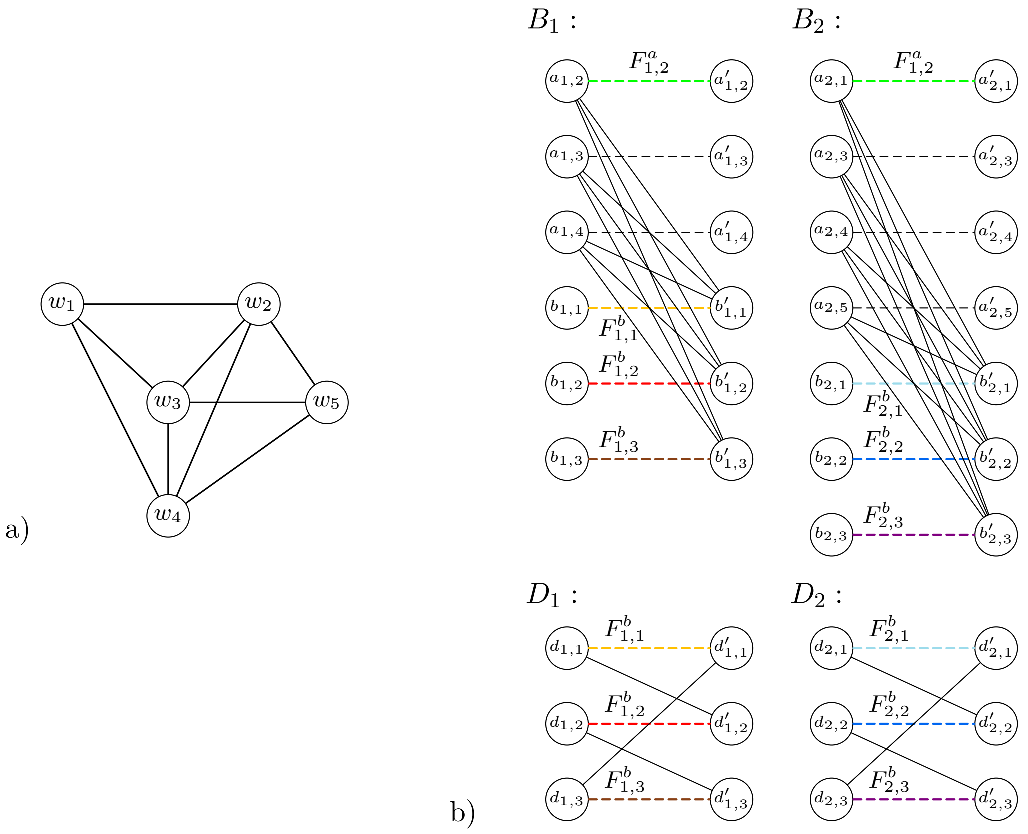

a)
w_1
w_2
w_3
w_4
w_5
b)
B_1 :
B_2 :
D_1 :
D_2 :
$F^a_{1,2}$
$F^b_{1,1}$
$F^b_{1,2}$
$F^b_{1,3}$
$F^b_{2,1}$
$F^b_{2,2}$
$F^b_{2,3}$
$a_{1,2}$ $a_{1,3}$ $a_{1,4}$ $b_{1,1}$ $b_{1,2}$ $b_{1,3}$
$a'_{1,2}$ $a'_{1,3}$ $a'_{1,4}$ $b'_{1,1}$ $b'_{1,2}$ $b'_{1,3}$
$a_{2,1}$ $a_{2,3}$ $a_{2,4}$ $a_{2,5}$ $b_{2,1}$ $b_{2,2}$ $b_{2,3}$
$a'_{2,1}$ $a'_{2,3}$ $a'_{2,4}$ $a'_{2,5}$ $b'_{2,1}$ $b'_{2,2}$ $b'_{2,3}$
$d_{1,1}$ $d_{1,2}$ $d_{1,3}$ $d'_{1,1}$ $d'_{1,2}$ $d'_{1,3}$
$d_{2,1}$ $d_{2,2}$ $d_{2,3}$ $d'_{2,1}$ $d'_{2,2}$ $d'_{2,3}$

Figure 5.2.: a) Example graph H. b) Auxiliary graphs $G_1 = B_1 + D_1$ and $G_2 = B_2 + D_2$, corresponding to nodes w_1, w_2 in H and parameter $l = 4$. For ease of notation we identify a node w_i with its index i. Equally colored (non-black) edges belong to the same couple, which is notated along them. For clearness, all on-level edges are shown dashed.

Proof. It is clear that the problem is in NP, as for any proposed solution $M \subseteq E$, checking constraints (5.2)-(5.6) and determining the size of M can be done in polynomial time.

In order to show that the CLCMP-L decision problem is NP-hard, the decision version of the clique problem will be polynomially reduced to it.

Let (H, l) be an instance of the clique decision problem, where $H = (W_H, E_H)$ is a graph with $W_H = \{w_1, \ldots, w_n\}$, $|E_H| = m$ and l is an integer with $0 \leq l \leq n$. The question to answer is whether H contains a clique $C \subseteq W_H$ of size at least l and, if it does, what it looks like. We will consider the equivalent question, whether H contains a clique of size exactly l.

Based on H and l, we will construct a problem instance of the CLCMP-L decision problem which consists of a bipartite level graph G', a couple collection $\mathcal{F}$ composed of on-level edges only, and integers k and l' serving as parameters for the level constraint and as a decision parameter, respectively.

We start by defining the graph $G' = (U \uplus V, E)$ using the introduced building blocks. We remark that whenever a node w_i appears as an index we replace it with i for ease of notation.

For each node w_i in W_H and the set of its neighbors, $\Gamma(w_i)$, let G_i be the auxiliary graph as constructed in Lemma 5.5. The graph $G' = (U \uplus V, E)$ is composed of these auxiliary graphs, i.e. $G' := G_1 + \ldots + G_n$. Each auxiliary graph G_i is of the form $G_i = B_i + D_i$, where $B_i = (X_i \uplus X_i', E_i^B)$ and $D_i = (Y_i \uplus Y_i', E_i^D)$ for all $i = 1, \ldots, n$. Of course, $|U| = |V|$ and G' is a bipartite level graph.

In order to determine the size of G', we add up the sizes of the auxiliary graphs, which are given in Lemma 5.5:

$$\begin{aligned} |U| = |V| &= \sum_{i=1}^{n} |U_i| \\ &= \sum_{i=1}^{n} (\deg_H(w_i) + 2(l-1)) \\ &= 2m + 2n(l-1), \end{aligned} \tag{5.7}$$

$$\begin{aligned} |E| &= \sum_{i=1}^{n} |E_i| \\ &= \sum_{i=1}^{n} (\deg_H(w_i) l + 3(l-1)) \\ &= 2ml + 3n(l-1). \end{aligned} \tag{5.8}$$

The graph G' is motivated by the following idea. For $i = 1, \ldots, n$, each node $a_{i,j} \in X_i$ stands for an edge $[w_i, w_j]$ emanating from node w_i in the graph H. In the subgraph B_i, each of the $l-1$ nodes $b'_{i,h}$ stand for the possibility to cover a node $a_{i,j}$ by an off-level edge. This represents the fact that if a node $w_i \in W_H$ is a member of a clique C of size l, it has $l-1$ neighbors in C. So, if an off-level edge $[a_{i,j}, b'_{i,h}]$ is in a matching for some $h \in \{1, \ldots, l-1\}$, this indicates that both end-nodes of $[w_i, w_j]$ are members of the clique C. On the other hand, if an on-level edge $[a_{i,j}, a'_{i,j}]$ is in a matching for some $[w_i, w_j] \in E_H$, this indicates that not both end-nodes of $[w_i, w_j]$ are members of the clique C.

One should note that for each edge $[w_i, w_j]$ in E_H there are two corresponding nodes $a_{i,j}$ and $a_{j,i}$ in the subgraphs B_i and B_j, respectively. Hence, there are $2m$ nodes of the form $a_{i,j}$ in G' in total.

The entire couple collection $\mathcal{F}$ comprises the couples $F^b_{i,h}$ for all $i = 1, \ldots, n$ and $h = 1, \ldots, l-1$ and the couples

$$F^a_{i,j} := \Big\{ [a_{i,j}, a'_{i,j}], [a_{j,i}, a'_{j,i}] \Big\} \quad \text{for all } [w_i, w_j] \in E_H.$$

Hence, $\mathcal{F}$ can be written as

$$\mathcal{F} = \{F^a_{i,j} \mid i = 1, \ldots, n; j : [w_i, w_j] \in E_H\} \cup \{F^b_{i,h} \mid i = 1, \ldots, n; h = 1, \ldots, l-1\}.$$

While for a fixed index i the couples in $F^b_{i,h}$ can be seen as links between the subgraphs B_i and D_i, the couples in $F^a_{i,j}$ connect the subgraphs B_i and B_j, for $i \neq j$. In Figure 5.2, two edges of the same couple are marked with the same color. The couple collection $\mathcal{F}$ consists of $p = m + n(l-1)$ couples. All couples contain on-level edges only. So, (after renaming the nodes in V_i^B and V_i^D) the couple constraints can be written in the form (5.4).

The next parameter to define is the integer k for the level constraint. As k states how many on-level edges of G' a feasible matching must contain, this constraint establishes a connection among all the subgraphs in G'. The parameter is defined as follows:

$$k := 2m + (2n - 3l)(l-1).$$

The decision parameter l' completes the problem instance of the CLCMP-L decision problem. It is defined as

$$l' := 2m + (2n - l)(l-1).$$

The construction of the problem instance $(G', \mathcal{F}, k, l')$ is polynomial in the input size of the instance $(H = (W_H, E_H), l)$. Properties (5.7) and (5.8) ensure that the graph G' is of polynomial size in $|W_H|, |E_H|$ and l. It is important to notice that l itself is bounded above by n. Furthermore, all steps during the definition of G' can be done in polynomial time. The same holds true for the definition of the couple collection $\mathcal{F}$, which is basically an ordered listing of the on-level edges in G'. The parameters k and l' can also be calculated in polynomial time.

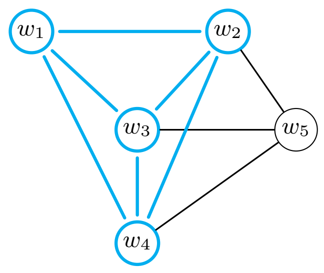
Next, we show that the following equivalence holds: The graph H contains a clique of size exactly l if and only if the graph G' contains a matching which meets constraints (5.2)-(5.6) and is of size at least l'.

So, let $H = (W_H, E_H)$ be a graph with $W_H = \{w_1, \ldots, w_n\}$, $|E_H| = m$ and let $C = \{w_{i_1}, \ldots, w_{i_l}\}$ be a clique in H. Let $G', \mathcal{F}, k$ and l' be defined as above. Based on C and l, a matching M in the graph G' will be defined next. An example of what M looks like for a graph H with a clique C and a parameter l can be found in Figure 5.3.

As mentioned before, an off-level edge in M, which covers a node a_{i_s,i_t}, indicates that the nodes w_{i_s} and w_{i_t} in H both are members of the clique C. In other words, it indicates that the edge $[w_{i_s}, w_{i_t}]$ is in $E_H(C)$. An on-level edge in M, which covers a node a_{i_s,i_t}, indicates that not both nodes w_{i_s} and w_{i_t} are members of the clique C. In other words, it indicates that the edge $[w_{i_s}, w_{i_t}]$ is not in $E_H(C)$. Anyway, it may still be the case that one of the nodes w_{i_s}, w_{i_t} is in the clique.

For a better understanding of how the matching M is composed, we partition M into several subsets. To this end, first we consider nodes w_{i_s} which are members of C. The decision as to which edges of the corresponding auxiliary graph G_{i_s} are in the matching depends on which neighboring nodes of w_{i_s} are also in C. We similarly consider the nodes w_q which are not in C.

We start by considering a node $w_{i_s} \in C$ and define which edges in the corresponding auxiliary graph G_{i_s} are in M. For each node $w_{i_t} \in C$ with $w_{i_t} \neq w_{i_s}$, the matching contains an off-level edge that is incident to the node a_{i_s,i_t} in the subgraph B_{i_s}. The set

a)


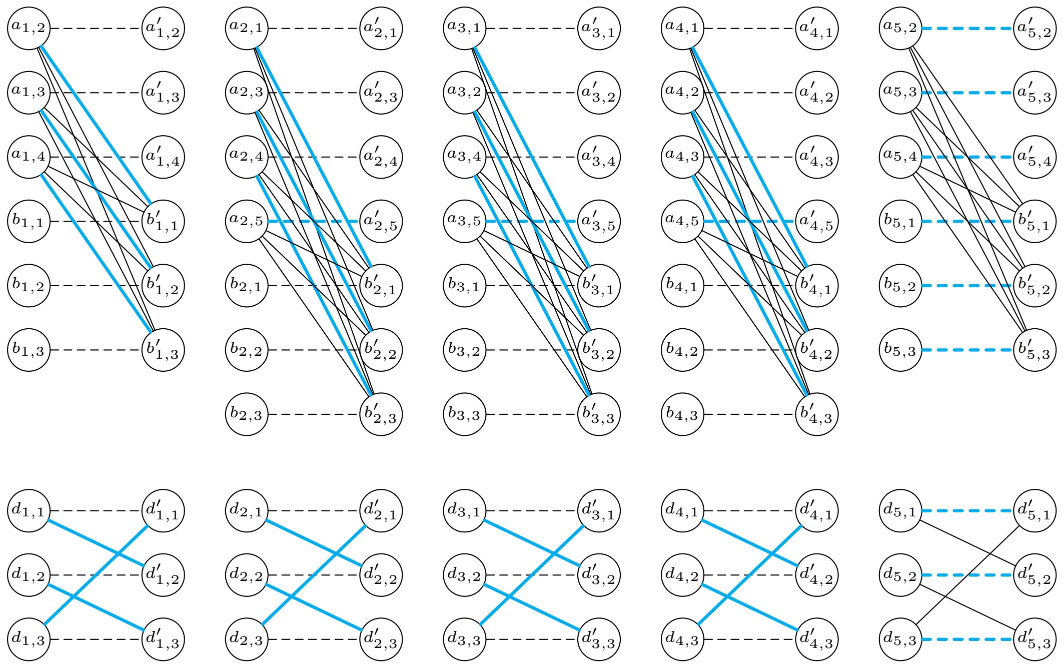
b)


Figure 5.3.: a) Example graph H with clique C of size $l = 4$. Nodes in C and edges in $E(C)$ are blue colored. b) Graph G' corresponding to H and l. The blue edges in G' build a matching M, which corresponds to C and satisfies (5.2) – (5.5).

containing these edges is denoted by $M_{i_s}^B$. For each node $w_q \notin C$ which is adjacent to w_{i_s} in H, the matching contains the on-level edge $[a_{i_s,q}, a'_{i_s,q}]$ in the subgraph B_{i_s}. The set containing these edges is denoted by $\bar{M}_{i_s}^B$. In the subgraph D_{i_s}, all off-level edges are defined to be in the matching. The set containing these edges is denoted by $M_{i_s}^D$. We apply these construction rules for all nodes $w_{i_s}, s = 1, \ldots, l$, and the corresponding auxiliary graphs $G_{i_s}, s = 1, \ldots, l$. The resulting subsets of M are as follows:

$$\begin{aligned}
M_{i_s}^B :=& \{[a_{i_s,i_t}, b'_{i_s,t}] \mid t = 1, \ldots, s-1\} && \\
& \cup \{[a_{i_s,i_t}, b'_{i_s,t-1}] \mid t = s+1, \ldots, l\} && \text{for all } s = 1, \ldots, l, \\
\bar{M}_{i_s}^B :=& \Big\{[a_{i_s,q}, a'_{i_s,q}] \mid \exists\ [w_{i_s}, w_q] \in E_H \text{ with } w_q \notin C\Big\} && \text{for all } s = 1, \ldots, l, \\
M_{i_s}^D :=& \{[d_{i_s,h}, d'_{i_s,h+1}] \mid h = 1, \ldots, l-1\} && \text{for all } s = 1, \ldots, l,
\end{aligned}$$

with $d_{i_s,l}$ being defined as $d_{i_s,1}$.

Now, we come to the auxiliary graphs G_q corresponding to the nodes $w_q \notin C$. Here, all on-level edges in both subgraphs B_q and D_q are contained in the matching. The sets containing these edges are denoted by $\bar{M}_q^B$ and $\bar{M}_q^D$, respectively. Again, we apply these construction rules for all nodes $w_q \notin C$ and the corresponding auxiliary graphs G_q. The resulting subsets of M are as follows:

$$\begin{aligned}
\bar{M}_q^B :=& \Big\{[a_{q,t}, a'_{q,t}] \mid \exists\ [w_q, w_t] \in E_H\Big\} && \\
& \cup \Big\{[b_{q,h}, b'_{q,h}] \mid h = 1, \ldots, l-1\Big\} && \text{for all } q \text{ with } w_q \notin C, \\
\bar{M}_q^D :=& \Big\{[d_{q,h}, d'_{q,h}] \mid h = 1, \ldots, l-1\Big\} && \text{for all } q \text{ with } w_q \notin C.
\end{aligned}$$

The matching M finally is composed of the above subsets:

$$M := \bigcup_{s=1}^{l} \left(M_{i_s}^B \cup \bar{M}_{i_s}^B \cup M_{i_s}^D\right) \cup \bigcup_{q: w_q \notin C} \left(\bar{M}_q^B \cup \bar{M}_q^D\right).$$

In the example in Figure 5.3, the corresponding sets for the indices $s = 2$ (with $i_2 = 2$) and $q = 5$ are:

$$\begin{aligned}
M_2^B &= \Big\{[a_{2,1}, b'_{2,1}], [a_{2,3}, b'_{2,2}], [a_{2,4}, b'_{2,3}]\Big\}, \\
\bar{M}_2^B &= \Big\{[a_{2,5}, a'_{2,5}]\Big\}, \\
M_2^D &= \Big\{[d_{2,1}, d'_{2,2}], [d_{2,2}, d'_{2,3}], [d_{2,3}, d'_{2,1}]\Big\}, \\
\bar{M}_5^B &= \Big\{[a_{5,j}, a'_{5,j}], [b_{5,h}, b'_{5,h}] \mid j = 2, 3, 4; h = 1, 2, 3\Big\}, \\
\bar{M}_5^D &= \Big\{[d_{5,1}, d'_{5,1}], [d_{5,2}, d'_{5,2}], [d_{5,3}, d'_{5,3}]\Big\}.
\end{aligned}$$

Concerning the cardinality of the above defined subsets of M, we have that

$$\begin{aligned}
|M_{i_s}^B| &= l-1 && \text{for all } s=1,\dots,l,\\
|\bar{M}_{i_s}^B| &= \deg_H(w_{i_s})-(l-1) && \text{for all } s=1,\dots,l,\\
|M_{i_s}^D| &= l-1 && \text{for all } s=1,\dots,l,\\
|\bar{M}_q^B| &= \deg_H(w_q)+l-1 && \text{for all } q: w_q\notin C,\\
|\bar{M}_q^D| &= l-1 && \text{for all } q: w_q\notin C.
\end{aligned}$$

We now verify the demanded properties of M. First of all, for $s=1,\dots,l$ and q with $w_q\notin C$, the sets $M_{i_s}^B, M_{i_s}^D, \bar{M}_{i_s}^B, \bar{M}_q^B$ and $\bar{M}_q^D$ are matchings. As the corresponding subgraphs, B_{i_s}, D_{i_s}, B_q and D_q, are pairwise node-disjoint, M is a matching as well, and therefore satisfies constraints (5.2) and (5.3).

We continue by checking the couple constraints. These constraints are affected by on-level edges only. The subsets of M, in which they appear, are the sets $\bar{M}_{i_s}^B, \bar{M}_q^B$ and $\bar{M}_q^D$ for all $s=1,\dots,l$ and q with $w_q\notin C$. So, we need to consider all edges of the form $[a_{i,j},a'_{i,j}]$, with at least one of the nodes w_i,w_j not being in C, and all edges of the form $[b_{q,h},b'_{q,h}]$ and $[d_{q,h},d'_{q,h}]$, with q such that $w_q\notin C$.

For any $s=1,\dots,l$, any q,t with $w_q,w_t\notin C$ and any $h=1,\dots,l-1$ it holds that

$$[a_{i_s,q},a'_{i_s,q}]\in\bar{M}_{i_s}^B \Leftrightarrow [a_{q,i_s},a'_{q,i_s}]\in\bar{M}_q^B, \tag{5.9}$$
$$[a_{q,t},a'_{q,t}]\in\bar{M}_q^B \Leftrightarrow [a_{t,q},a'_{t,q}]\in\bar{M}_t^B, \tag{5.10}$$
$$[b_{q,h},b'_{q,h}]\in\bar{M}_q^B \Leftrightarrow [d_{q,h},d'_{q,h}]\in\bar{M}_q^D. \tag{5.11}$$

Equivalences (5.9) and (5.10) state that the couple constraints concerning the couples in $F_{i,j}^a$ hold for all $[w_i,w_j]\in E_H$. The equivalences in (5.11) ensure that the couple constraints concerning the couples in $F_{i,h}^b$ are satisfied for all $i=1,\dots,n$ and $h=1,\dots,l-1$. Hence, the couple constraints (5.4) are fulfilled.

Next, the level constraint is verified. For that, the cardinalities of the sets $\bigcup_{s=1}^l \bar{M}_{i_s}^B$, $\bigcup_{q:w_q\notin C}\bar{M}_q^B$ and $\bigcup_{q:w_q\notin C}\bar{M}_q^D$ are added. We have that

$$\begin{aligned}
&|\bigcup_{s=1}^{l}\bar{M}_{i_s}^B| + |\bigcup_{q:w_q\notin C}\bar{M}_q^B| + |\bigcup_{q:w_q\notin C}\bar{M}_q^D|\\
&= \sum_{w_i\in C}(\deg_H(w_i)-(l-1)) + \sum_{w_q\in W_H\setminus C}(\deg_H(w_q)+l-1) + \sum_{w_q\in W_H\setminus C}(l-1)\\
&= \sum_{w_i\in W_H}\deg_H(w_i) - \sum_{w_i\in C}(l-1) + \sum_{w_q\in W_H\setminus C}(l-1) + \sum_{w_q\in W_H\setminus C}(l-1)\\
&= 2m - l(l-1) + 2(n-l)(l-1)\\
&= 2m + (2n-3l)(l-1).
\end{aligned}\tag{5.12}$$

Hence, the level constraint (5.5) is fulfilled.

The total size of M can be obtained easily now. The edges in M are either on-level edges, or they are in the sets $M_{i_s}^B$ or $M_{i_s}^D$, $i=1,\dots,l$. The number of on-level edges is given in (5.12). Concerning the cardinalities of the latter sets we have that $|\bigcup_{s=1}^{l} M_{i_s}^{B}| = |\bigcup_{s=1}^{l} M_{i_s}^{D}| = l(l-1)$. So, the cardinality of M is

$$|M| = 2m + (2n - 3l)(l - 1) + 2l(l - 1) = 2m + (2n - l)(l - 1).$$

As $|M| \geq l'$, the matching M proves $(G', \mathcal{F}, k, l')$ to be a "yes"-instance of the CLCMP-L decision problem.

Now we assume that the graph $H = (W_H, E_H)$ with $W_H = \{w_1, \dots, w_n\}$ and $|E_H| = m$ has no clique of size l. Let $G', \mathcal{F}, k, l'$ be defined as above, based on H and l. We show that there is no matching in G', which fulfills the couple constraints for $\mathcal{F}$, the level constraint with parameter k and is of size at least l'. Figure 5.4 illustrates this case for an example graph H.

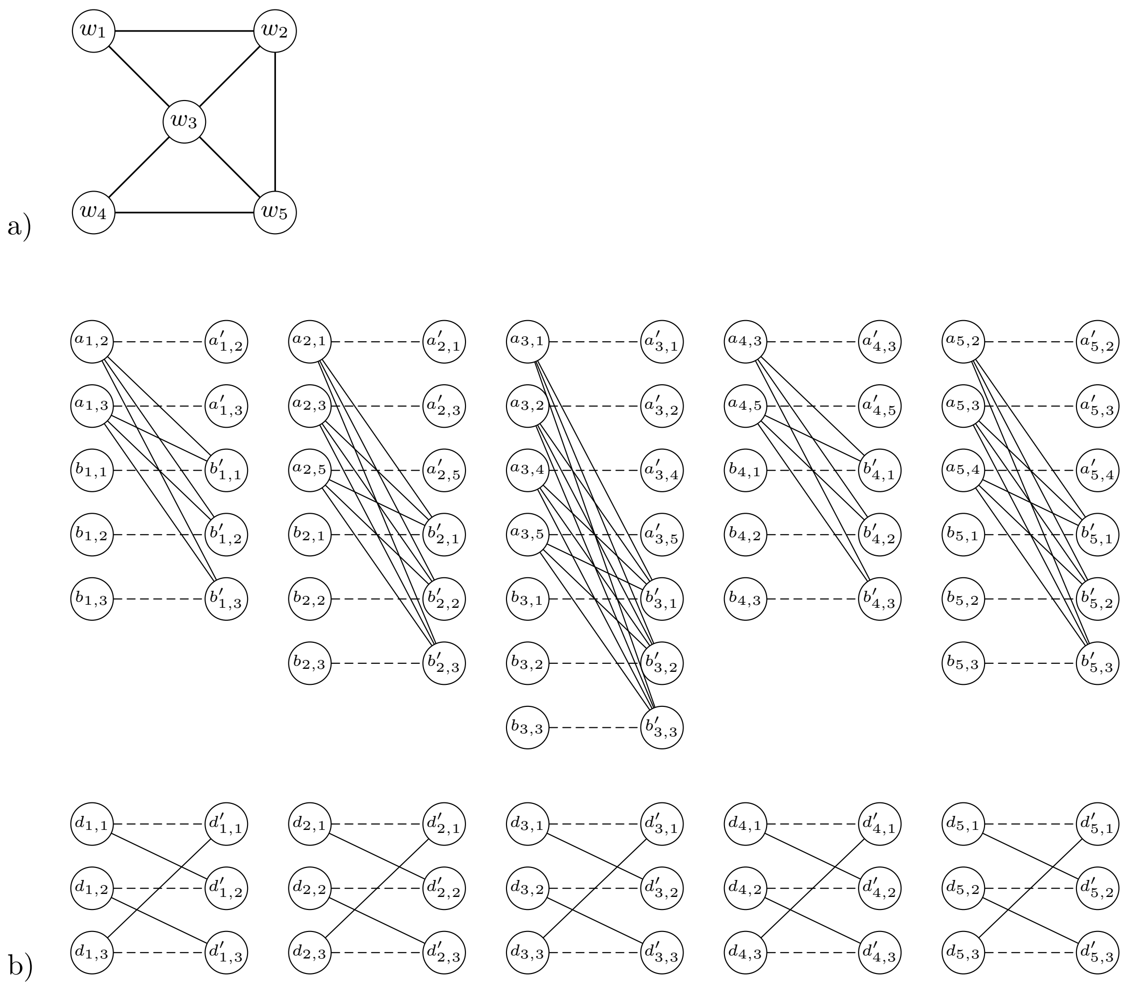

a)
w_1
w_2
w_3
w_4
w_5
b)
$a_{1,2}$ $a'_{1,2}$ $a_{1,3}$ $a'_{1,3}$ $b_{1,1}$ $b'_{1,1}$ $b_{1,2}$ $b'_{1,2}$ $b_{1,3}$ $b'_{1,3}$
$a_{2,1}$ $a'_{2,1}$ $a_{2,3}$ $a'_{2,3}$ $a_{2,5}$ $a'_{2,5}$ $b_{2,1}$ $b'_{2,1}$ $b_{2,2}$ $b'_{2,2}$ $b_{2,3}$ $b'_{2,3}$
$a_{3,1}$ $a'_{3,1}$ $a_{3,2}$ $a'_{3,2}$ $a_{3,4}$ $a'_{3,4}$ $a_{3,5}$ $a'_{3,5}$ $b_{3,1}$ $b'_{3,1}$ $b_{3,2}$ $b'_{3,2}$ $b_{3,3}$ $b'_{3,3}$
$a_{4,3}$ $a'_{4,3}$ $a_{4,5}$ $a'_{4,5}$ $b_{4,1}$ $b'_{4,1}$ $b_{4,2}$ $b'_{4,2}$ $b_{4,3}$ $b'_{4,3}$
$a_{5,2}$ $a'_{5,2}$ $a_{5,3}$ $a'_{5,3}$ $a_{5,4}$ $a'_{5,4}$ $b_{5,1}$ $b'_{5,1}$ $b_{5,2}$ $b'_{5,2}$ $b_{5,3}$ $b'_{5,3}$
$d_{1,1}$ $d'_{1,1}$ $d_{1,2}$ $d'_{1,2}$ $d_{1,3}$ $d'_{1,3}$
$d_{2,1}$ $d'_{2,1}$ $d_{2,2}$ $d'_{2,2}$ $d_{2,3}$ $d'_{2,3}$
$d_{3,1}$ $d'_{3,1}$ $d_{3,2}$ $d'_{3,2}$ $d_{3,3}$ $d'_{3,3}$
$d_{4,1}$ $d'_{4,1}$ $d_{4,2}$ $d'_{4,2}$ $d_{4,3}$ $d'_{4,3}$
$d_{5,1}$ $d'_{5,1}$ $d_{5,2}$ $d'_{5,2}$ $d_{5,3}$ $d'_{5,3}$

Figure 5.4.: a) Example graph H which does not contain a clique of size $l = 4$. b) Graph G' resulting from problem reduction. There is no matching in G' which satisfies (5.2) – (5.5) and which is of size at least $l' = 32$.

Let M be a matching in the graph G', which fulfills the couple constraints concerning $\mathcal{F}$ and satisfies the level constraint for parameter k. In order to obtain information about the cardinality of M, let α be the total number of the off-level edges in M which are in the subgraphs $B_1, \ldots, B_n$. We call these edges *alpha-edges*. Let us consider any off-level edge $[a_{i,j}, b'_{i,h}]$ in the subgraph B_i, for some $i = 1, \ldots, n$ and $h = 1, \ldots, l-1$. If this edge is in M, none of the two on-level edges $[a_{i,j}, a'_{i,j}]$ and $[b_{i,h}, b'_{i,h}]$ can be in M. The structure of the subgraph B_i ensures that each of its on-level edges shares exactly one end-node with an off-level edge. Due to this fact, it holds that if $[a_{i,j}, b'_{i,h}]$ is in M, then it will be the only off-level edge in M sharing an end-node with $[a_{i,j}, a'_{i,j}]$ or $[b_{i,h}, b'_{i,h}]$. As a consequence, the alpha-edges in M prevent 2α on-level edges in $B_1, \ldots, B_n$ from being in M. Additionally, when an on-level edge $[b_{i,h}, b'_{i,h}]$ is prevented from being in the matching M, the couple constraints concerning the couple $F^b_{i,h}$ ensure that the edge $[d_{i,h}, d'_{i,h}]$ in the subgraph D_i is not in M, either. Hence, the alpha-edges in M prevent 3α on-level edges in G' from being in M.

M is assumed to contain $k = 2m + (2n-3l)(l-1)$ on-level edges. According to (5.7), there are $2m + 2n(l-1)$ on-level edges in G' in total. So, the following estimation is valid:

$$\begin{aligned} & 2m + 2n(l-1) - 3\alpha \geq 2m + (2n-3l)(l-1) \\ \Leftrightarrow \quad & \alpha \leq l(l-1). \end{aligned} \tag{5.13}$$

Let β be the total number of the off-level edges in M which are in the subgraphs $D_1, \ldots, D_n$. We call these edges *beta-edges*. These edges prevent at least β on-level edges in the subgraphs $D_1, \ldots, D_n$ from being in M. Additionally, when an on-level edge $[d_{i,h}, d'_{i,h}]$ is prevented from being in the matching M, the couple constraints concerning couple $F^b_{i,h}$ ensure that the corresponding edge $[b_{i,h}, b'_{i,h}]$ in the subgraph B_i is not in M, either.

The consequences of all alpha-edges and beta-edges in M are combined in the following way. For each alpha-edge $[a_{i,j}, b'_{i,h}]$ in M, we take into account that the on-level edge $[a_{i,j}, a'_{i,j}]$ cannot be in M. For each beta-edge $[d_{i,h}, d'_{i,h+1}]$ in M, we take into account that both on-level edges $[d_{i,h}, d'_{i,h}]$ and $[b_{i,h}, b'_{i,h}]$ cannot be in M.

Again, as the level constraint is supposed to be fulfilled, the following inequality must be valid:

$$\begin{aligned} & 2m + 2n(l-1) - \alpha - 2\beta \geq 2m + (2n-3l)(l-1) \\ \Leftrightarrow \quad & \alpha + \beta \leq \frac{1}{2}\alpha + \frac{3}{2}l(l-1). \end{aligned} \tag{5.14}$$

Together with the upper bound for α from (5.13), it must hold that

$$\alpha + \beta \leq 2l(l-1).$$

As all edges in the matching M are either on-level or off-level edges in G', we have that $|M| = k + \alpha + \beta$. If M is assumed to be of size at least l', it follows that

$$\begin{aligned} & 2m + (2n-3l)(l-1) + \alpha + \beta \geq 2m + (2n-l)(l-1) \\ \Leftrightarrow \quad & \alpha + \beta \geq 2l(l-1). \end{aligned}$$

This leads to the fact that $|M| \geq l'$ if and only if $\alpha + \beta = 2l(l-1)$.

There are three more direct consequences of inequality (5.14):

Claim. *If M fulfills the level constraint and $|M| \geq l'$, the following statements hold:*

1. *$\beta \leq l(l-1)$.*
2. *Each node of the form $a_{i,j}$ and each node of the form $d_{i,h}$ in G' is covered by an edge in M.*
3. *The beta-edges in M are distributed as follows: Each subgraph $D_i, i = 1, \ldots, n$, has either none or all of its off-level edges in M.*

Proof of Claim. 1. Seeking a contradiction, let us assume that β can be written as $\beta = l(l-1) + \delta$, with $\delta \geq 1$. Applying this value of β to inequality 5.14 implies that $\alpha \leq l(l-1) - 2\delta$. This contradicts $\alpha + \beta = 2l(l-1)$.

2. Let γ and δ denote the number of nodes of the form $a_{i,j}$ and $d_{i,h}$, respectively, which are not covered by an edge in M. Then, none of the on-level edges incident to any of these nodes is in M, while none of the off-level edges incident to them is in M, either. Hence, none of the on-level edges incident to these nodes is prevented from being in M by an alpha-edge or beta-edge. Taking this into account in estimation (5.14), the following inequality must be valid:

$$\begin{aligned} & 2m + 2n(l-1) - \alpha - 2\beta - \gamma - \delta \geq 2m + (2n-3l)(l-1) \\ \Leftrightarrow \quad & \alpha + \beta \leq 2l(l-1) - \frac{1}{2}\gamma - \frac{1}{2}\delta. \end{aligned}$$

Due to the fact that $\alpha + \beta = 2l(l-1)$ this implies that $\gamma = \delta = 0$.

3. Otherwise, in each subgraph D_i where this is not the case, the number of its on-level edges prevented from being in M is strictly greater than the number of its off-level edges in M. Then, inequality (5.14) can be strengthened to

$$\begin{aligned} & 2m + 2n(l-1) - \alpha - 2\beta - 1 \geq 2m + (2n-3l)(l-1) \\ \Leftrightarrow \quad & \alpha + \beta \leq 2l(l-1) - \frac{1}{2}, \end{aligned}$$

which contradicts $\alpha + \beta = 2l(l-1)$. $\square$

The bounds $\alpha \leq l(l-1), \beta \leq l(l-1)$ together with equality $\alpha + \beta = 2l(l-1)$ imply that

$$\alpha = \beta = l(l-1).$$

Now we can continue from part 3 of the claim. As $\beta = l(l-1)$, there are exactly l subgraphs among $D_1, \ldots, D_n$ which have exactly $l-1$ off-level edges in M. All other subgraphs among $D_1, \ldots, D_n$ do not contain any off-level edges in M, but all of their on-level edges are contained in M due to part 2 of the claim.

Furthermore, the couple constraints concerning the couples $F^b_{i,h}$ for all $i = 1, \dots, n$ and $h = 1, \dots, l-1$, imply that exactly l of the subgraphs $B_i, i = 1, \dots, n$, have no on-level edges of the form $[b_{i,h}, b'_{i,h}]$ in M, while the other subgraphs have all on-level edges of the form $[b_{i,h}, b'_{i,h}]$ in M. As $\alpha = l(l-1)$, there are exactly l subgraphs among $B_1, \dots, B_n$ which have exactly $l-1$ off-level edges in M. Let $B_{i_1}, \dots, B_{i_l}$ be those subgraphs. If node $a_{i_s,j}$ in B_{i_s} is covered by an off-level edge in M, then, due to the couple constraints concerning couple $F^a_{i_s,j}$ and due to part 2 of the claim, node a_{j,i_s} in B_j is covered by an off-level edge as well. Hence, $B_j \in \{B_{i_s} \mid s = 1, \dots, l\}$. So, all nodes $a_{i,j}$ which are covered by an off-level edge in M are the nodes a_{i_s,i_t} for all $s, t = 1, \dots, l$ with $s \neq t$. The existence of these nodes implies that the edge $[w_{i_s}, w_{i_t}]$ exists in H for all $s, t = 1, \dots, l$ with $s \neq t$. Hence, $\{w_{i_1}, \dots, w_{i_l}\}$ is a clique in H. This contradicts the assumption that there is no clique of size l in H.

In conclusion, if the graph H does not contain a clique of size l, then there is no matching in G' which fulfills the couple constraints concerning $\mathcal{F}$, satisfies the k-cardinality constraint and is of cardinality at least l'. $\square$

6. Properties of Resource Constrained Matching Polytopes

The polytopal structure of a resource constrained matching problem is interesting not only from a theoretical point of view but also for the development of solution methods.

Aboudi and Nemhauser [3] present a class of facet inducing inequalities of the polytope of the CCAP-L. We recapitulate their results in Section 6.1.

In Section 6.2, we establish a characterization of the non-integral vertices of the polytope corresponding to the linear relaxation of the LCPMP. For each non-integral vertex $\bar{x}$ of the polytope of the relaxed problem, we present a set of inequalities that separate $\bar{x}$ from the convex hull of integral solutions. Furthermore, we show that for any given non-integral vertex a specific inequality that separates the vertex from the convex hull of integer solutions can be determined in polynomial time.

6.1. Some facets for the couple constrained assignment polytope

In [3], Aboudi and Nemhauser introduce a class of facets for the polytope of the couple constrained assignment problem with on-level couples. For presenting this class of facets we recall the constraints of this problem as introduced in Section 3.3:

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad \forall\ i = 1, \dots, n \tag{6.1}$$

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad \forall\ j = 1, \dots, n \tag{6.2}$$

$$x_{2r-1,2r-1} = x_{2r,2r} \qquad \forall\ r = 1, \dots, p \tag{6.3}$$

$$x_{ij} \in \{0,1\} \qquad \forall\ i, j = 1, \dots, n, \tag{6.4}$$

where the constraints in (6.3) are the couple constraints concerning the couple collection $\mathcal{F} = \{F_1, \dots, F_p\}$ with $F_r = \{(2r-1, 2r-1), (2r, 2r)\}$ for all $r = 1, \dots, p$.

Let $S = \left\{x \in \{0,1\}^{n^2} \mid x \text{ satisfies } (6.1) - (6.4)\right\}$ be the set of feasible solutions to the CCAP-L. The *couple constrained assignment polytope* – for the case of on-level couples – is the convex hull of S, denoted by $\operatorname{conv}(S)$. Further, let the polytope corresponding to the linear relaxation of this problem be $P = \left\{x \in \mathbb{R}^{n^2}_{\geq 0} \mid x \text{ satisfies } (6.1) - (6.3)\right\}$.

In order to be compliant with the notation of Aboudi and Nemhauser, let I and J be arbitrary subsets of $N = \{1, \dots, n\}$. Further, let $x(I, J) = \sum_{i \in I} \sum_{j \in J} x_{ij}$. We define the sets K and $\tilde{K}$, which will appear in the facet inducing inequalities, as follows:

$$\begin{aligned}K &:= \{2r-1 \mid \{2r-1, 2r\} \subseteq I \cap J, r = 1, \dots, p\}\,,\\ \tilde{K} &:= \{2r-1 \mid \{2r-1, 2r\} \subseteq (N \setminus I) \cap (N \setminus J), r = 1, \dots, p\}\,.\end{aligned}$$

Aboudi and Nemhauser show in [3] that the following holds true:

Theorem 6.1. *Let $I, J \subseteq N$ with $|I| + |J| = n - 1$. Let $|\tilde{K}| \geq 1$. Then, the inequality*

$$\sum_{i \in K \cup \tilde{K}} x_{ii} \leq x(I, J) \tag{6.5}$$

defines a facet for conv(S).

Further, they show that in the special case of a single couple constraint $x_{11} = x_{22}$, the facets from Theorem 6.1 completely describe conv(S).

Theorem 6.2. *Let Q be the set of points satisfying inequality (6.5) for the CCAP-L with a single couple constraint:*

$$Q = \left\{x \in \mathbb{R}^{n^2} \mid x_{11} \leq x(I, J) \text{ for all } I, J \subseteq \{3, 4, \dots, n\} : I, J \neq \emptyset, |I| + |J| = n - 1\right\}.$$

Then, $\operatorname{conv}(S) = P \cap Q$.

6.2. Characterization of non-integral vertices of the level constrained perfect matching polytope

In this section, we establish a class of inequalities which have the property to separate all non-integral vertices of the LP-relaxation of the LCPMP from the convex hull of its integer solutions. For any such non-integral vertex $\bar{x}$ we show how to polynomially determine an inequality which separates $\bar{x}$ this way.

A problem strongly related to the LCPMP is the ECAP. In the ECAP there is an additional equality constraint which refers to a set of edges R that is not restricted to be the set of on-level edges. Further, the underlying graph in the ECAP is a complete bipartite graph. Alfakih et. al [7] present two classes of facet inducing inequalities for the polytope of the ECAP for instances belonging to what they call the *partitioned case*. It can be shown that if R is the set of on-level edges, then no problem instance of the ECAP with $n > 2$ belongs to the partitioned case. Hence, their results cannot be applied to the polytope of the LCPMP even when the underlying graph is complete.

We consider an LCPMP on the graph $G = (U \uplus V, E)$, with $U = \{u_1, \dots, u_n\}$ and $V = \{v_1, \dots, v_n\}$. The parameter of the level constraint is a nonnegative integer k with $k \leq n$. Let E_{on} denote the set of on-level edges in E. Due to the definition of the LCPMP in Section 2.2.2, each feasible solution to the LCPMP satisfies the following constraints:

$$\sum_{e\in\delta(u_i)} x_e = 1 \qquad \forall\, i = 1,\dots,n \tag{6.6}$$

$$\sum_{e\in\delta(v_i)} x_e = 1 \qquad \forall\, i = 1,\dots,n \tag{6.7}$$

$$\sum_{i=1}^{n} x_{[u_i,v_i]} = k \tag{6.8}$$

$$x_e \in \{0,1\} \qquad \forall\, e \in E. \tag{6.9}$$

Let $S = \left\{x \in \{0,1\}^E \mid x \text{ satisfies } (6.6) - (6.9)\right\}$ be the set of feasible solutions to the LCPMP. The *level constrained perfect matching polytope* is the convex hull of S. Further, let $P = \left\{x \in \mathbb{R}^E_{\geq 0} \mid x \text{ satisfies } (6.6) - (6.8)\right\}$ be the polytope corresponding to the linear relaxation of the LCPMP.

In order to establish our results, we take a closer look at the on-level edges contained in cycles in G, and the position at which they appear in the cycles.

Definition 6.3. *Let $G = (U \cup V, E)$ be a bipartite graph and let*

$$C = [z_1, e_{i_1}, z_2, \dots, z_{l-1}, e_{i_{l-1}}, z_l, e_{i_l}, z_1]$$

be a cycle in G, with $z_j \in U \cup V$ and $e_{i_j} \in E$ for all $j = 1,\dots,l$. Let C_{on} denote the set of all on-level edges in C and let C_{off} denote the set of all off-level edges in C.

We define $\mathcal{O}_C$ and $\mathcal{E}_C$ to be the sets of on-level edges which appear at an odd and at an even position in C, respectively:

$$\mathcal{O}_C := \left\{e_{i_j} \in C_{on} \mid j \text{ is odd}\right\},$$
$$\mathcal{E}_C := \left\{e_{i_j} \in C_{on} \mid j \text{ is even}\right\}.$$

We call $(\mathcal{O}_C, \mathcal{E}_C)$ a parity on-level partition *of C. A cycle C is called* balanced *if $|\mathcal{O}_C| = |\mathcal{E}_C|$. Otherwise, it is called* unbalanced.

See Figure 6.1 for a parity on-level partition of an example cycle. Obviously, a parity on-level partition of a cycle is not unique. It depends on which node in the cycle is considered as node z_1 and the direction in which the cycle is noted. So, for one cycle C there can be up to two parity on-level partitions which differ from each other only in that the sets $\mathcal{O}_C$ and $\mathcal{E}_C$ are interchanged.

Lemma 6.4. *Let $x \in P$ and let $C = [z_1, e_{i_1}, z_2, \dots, z_{l-1}, e_{i_{l-1}}, z_l, e_{i_l}, z_1]$ be a cycle in G, such that $\sum_{e\in E(C)} x_e = \frac{|C|}{2}$. Let $(\mathcal{O}_C, \mathcal{E}_C)$ be a parity on-level partition of C. Then the following holds:*

- *If C is balanced, then*

$$x(C_{on}) = |\mathcal{O}_C| = |\mathcal{E}_C|.$$

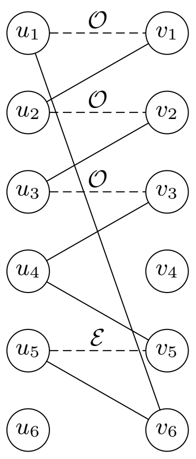


Figure 6.1.: Example of a bipartite cycle C and a parity on-level partition $(\mathcal{O}, \mathcal{E})$ of C with $|\mathcal{O}| = 3$ and $|\mathcal{E}| = 1$. All on-level edges of C are dashed. Above each on-level edge it is noted to which of the two sets $\mathcal{O}_C$ or $\mathcal{E}_C$ it belongs.

- *If x_e is integral for all $e \in E(C)$, then*

$$x(C_{on}) \in \{|\mathcal{O}_C|, |\mathcal{E}_C|\}.$$

- *If C is unbalanced and $0 < x_e < 1$ for all $e \in E(C)$, then*

$$\min\{|\mathcal{O}_C|, |\mathcal{E}_C|\} < x(C_{on}) < \max\{|\mathcal{O}_C|, |\mathcal{E}_C|\}.$$

Proof. Let x and C be as assumed. As $\sum_{e \in E(C)} x_e = \frac{|C|}{2}$ and x fulfills constraints (6.6) and (6.7), it holds that $x_{e_{i_j}} + x_{e_{i_{j+1}}} = 1$ for all $j = 1, \dots, l$, with $e_{i_{l+1}}$ being identified with e_{i_1}. Let $p := x_{e_{i_1}}$ and $q := x_{e_{i_2}}$. Then, we have that

$$\begin{aligned} x(\mathcal{O}_C) &= p|\mathcal{O}_C|, \\ x(\mathcal{E}_C) &= q|\mathcal{E}_C|. \end{aligned}$$

This has the following implications to $x(C_{\text{on}})$:

$$\begin{aligned} x(C_{\text{on}}) &= x(\mathcal{O}_C) + x(\mathcal{E}_C) \\ &= p|\mathcal{O}_C| + q|\mathcal{E}_C| \\ &\le p \max\{|\mathcal{O}_C|, |\mathcal{E}_C|\} + q \max\{|\mathcal{O}_C|, |\mathcal{E}_C|\} \\ &= \max\{|\mathcal{O}_C|, |\mathcal{E}_C|\}. \end{aligned} \tag{6.10}$$

Analogously, one can show that

$$x(C_{\text{on}}) \ge \min\{|\mathcal{O}_C|, |\mathcal{E}_C|\}. \tag{6.11}$$

If C is balanced, the bounds (6.10) and (6.11) imply that $x(C_{\text{on}}) = |\mathcal{O}_C| = |\mathcal{E}_C|$. If x_e is integral for all $e \in E(C)$, then either $p = 1$ and $q = 0$ or $p = 0$ and $q = 1$. Hence, $x(C_{\text{on}})$ is equal to either $|\mathcal{O}_C|$ or $|\mathcal{E}_C|$. If C is unbalanced and $0 < x_e < 1$ for all $e \in E(C)$, then the "$\leq$" sign in estimation (6.10) can be replaced by "$<$" and the "$\geq$" sign in estimation (6.11) can be replaced by "$>$". This completes the proof. □

For the special case where x is integral, it holds that $x_e = 1$ either for all edges $e \in \mathcal{O}_C$ or for all edges $e \in \mathcal{E}_C$. This is stated in Lemma 6.6. In its proof we make use of the following definition:

Definition 6.5. *Let $G = (U, E)$ be a graph. Let $P = [u_{i_1}, e_{i_1}, u_{i_2}, \ldots, u_{i_p}]$ be a path in G with $u_{i_h} \in U$ for all $h = 1, \ldots, p$ and $e_{i_h} \in E$ for all $h = 1, \ldots, p-1$. Let $Q = [u_{j_1}, e_{j_1}, u_{j_2}, \ldots, u_{j_q}]$ be a path in G with $u_{j_h} \in U$ for all $h = 1, \ldots, q$ and $e_{j_h} \in E$ for all $h = 1, \ldots, q-1$. Further, let $u_{i_p} = u_{j_1}$.*
We define the composition *of P and Q as*

$$P \circ Q := [u_{i_1}, e_{i_1}, u_{i_2}, \ldots, u_{i_p}, e_{j_1}, u_{j_2}, \ldots, u_{j_q}].$$

Lemma 6.6. *Let $x \in S$ and let C be an unbalanced cycle in G such that $x(E(C)) = \frac{|C|}{2}$. Let $(\mathcal{O}_C, \mathcal{E}_C)$ be a parity on-level partition of C. Then, exactly one of the following two statements holds true:*

- *$x(\mathcal{O}_C) = |\mathcal{O}_C|$ and $x(\mathcal{E}_C) = 0$;*
- *$x(\mathcal{O}_C) = 0$ and $x(\mathcal{E}_C) = |\mathcal{E}_C|$.*

Proof. We start with a general fact concerning the components of x which belong to a path of even length in the cycle C.

Claim. *For each path P of even length in the cycle C it holds that $x(E(P)) = \frac{|P|}{2}$.*

Proof of Claim. First, we show that for each path $\tilde{P}$ of even length in the graph G it holds that $x(E(\tilde{P})) \leq \frac{|\tilde{P}|}{2}$. Let $N(\tilde{P})$ denote the set of nodes contained in $\tilde{P}$. Constraints (6.6) and (6.7) imply that

$$|\tilde{P}| + 1 = \sum_{z \in N(\tilde{P})} x(\delta(z)) \geq 2 \sum_{e \in E(\tilde{P})} x_e = 2x(E(\tilde{P})).$$

Hence, $x(E(\tilde{P})) \leq \frac{|\tilde{P}|+1}{2}$. As x is integral and $|\tilde{P}|$ is even, this finally implies that $x(E(\tilde{P})) \leq \frac{|\tilde{P}|}{2}$. Since the path P is a path of even length in C and C is a cycle in G, it holds that $x(E(P)) \leq \frac{|P|}{2}$.

Now we show that $x(E(P)) \geq \frac{|P|}{2}$. Let P' be a path in C such that $P \circ P' = C$. Then,

$$\frac{|C|}{2} = x(E(C)) = \sum_{e \in E(P)} x_e + \sum_{e \in E(P')} x_e.$$

As P' is a path of even length in G as well, this implies that

$$x(E(P)) = x(E(C)) - x(E(P')) \geq \frac{|C|}{2} - \frac{|P'|}{2} = \frac{|P|}{2}.$$

Thus, $x(E(P)) = \frac{|P|}{2}$. □

Note that at least one of the sets $\mathcal{O}_C$ and $\mathcal{E}_C$ is nonempty, as C is unbalanced. Thus, at most one of the two statements in Lemma 6.6 holds true, as $|\mathcal{O}_C| \neq 0$ if $\mathcal{O}_C$ is nonempty and $|\mathcal{E}_C| \neq 0$ if $\mathcal{E}_C$ is nonempty.

Due to the fact that $x \in S$ and $x(E(C)) = \frac{|C|}{2}$, one of the values $x_{e_{i_j}}$ and $x_{e_{i_{j+1}}}$ is 1 and the other one is 0 for all edges e_{i_j} and $e_{i_{j+1}}$ appearing consecutively in C. This implies that if one of the sets $\mathcal{O}_C$ or $\mathcal{E}_C$ is empty, then the nonempty of the two sets (denoted by $\mathcal{B}$) satisfies either $x(\mathcal{B}) = |\mathcal{B}|$ or $x(\mathcal{B}) = 0$. For the empty of the two sets (denoted by $\bar{\mathcal{B}}$) it holds that $x(\bar{\mathcal{B}}) = |\bar{\mathcal{B}}| = 0$. Thus, in this special case the proof is complete. So, for the rest of the proof we assume that both sets $\mathcal{O}_C$ and $\mathcal{E}_C$ are nonempty.

We now show that for at least one of the sets $\mathcal{O}_C$ and $\mathcal{E}_C$ it holds that $x(\mathcal{O}_C) = 0$ or $x(\mathcal{E}_C) = 0$. To this end we assume that there exist two edges $e = [u_{i_1}, v_{j_1}] \in \mathcal{O}_C$ and $f = [u_{i_2}, v_{j_2}] \in \mathcal{E}_C$ with $x_e = 1$ and $x_f = 1$, and leading this assumption to a contradiction. Let $P_{e,f}$ denote the shortest path in C from an end-node of e to an end-node of f. Let $p := |P_{e,f}|$ be the length of that path. As $e \in \mathcal{O}_C$ and $f \in \mathcal{E}_C$, the path $P_{e,f}$ must be of even length. Due to the claim, it holds that $x\left(E(P_{e,f})\right) = \frac{|P_{e,f}|}{2}$. We can write $P_{e,f}$ as

$$P_{e,f} = [z_{i_1}, e_{i_1}, z_{i_2}, \ldots, z_{i_p}, e_{i_p}, z_{i_{p+1}}],$$

with $z_{i_1} \in \{u_{i_1}, v_{j_1}\}$ and $z_{i_{p+1}} \in \{u_{i_2}, v_{j_2}\}$. Let

$$P'_{e,f} = [z_{i_2}, e_{i_2}, z_{i_3}, \ldots, z_{i_{p-1}}, e_{i_{p-1}}, z_{i_p}]$$

be the path $P_{e,f}$ without the first and the last edge.

As $x_e = x_f = 1$, it holds that $x_{e_{i_1}} = x_{e_{i_p}} = 0$. As $P'_{e,f}$ is a path of even length in C, the claim implies

$$\frac{|P_{e,f}|}{2} = x\left(E(P_{e,f})\right) = x_{e_{i_1}} + x_{e_{i_p}} + x\left(E(P'_{e,f})\right) = \frac{|P'_{e,f}|}{2} = \frac{|P_{e,f}| - 2}{2}, \tag{6.12}$$

which is a contradiction. Hence, either $x(\mathcal{O}_C) = 0$ or $x(\mathcal{E}_C) = 0$.

Without loss of generality, let $x(\mathcal{E}_C) = 0$ (otherwise, i.e. in the that case $x(\mathcal{O}_C) = 0$, all edge indices in C can be increased by 1 in order to get a parity on-level partition of C with $\mathcal{E}_C$ and $\mathcal{O}_C$ switched).

To finish the proof, we now show that $x(\mathcal{O}_C) = |\mathcal{O}_C|$. To this end, we assume that $x(\mathcal{O}_C) < |\mathcal{O}_C|$. Let $e^* \in \mathcal{O}_C$ with $x_{e^*} = 0$ and let f^* be any edge in $\mathcal{E}_C$. Let P be the shortest path in C which contains the edges e^* and f^*. As $e^* \in \mathcal{O}_C$ and $f^* \in \mathcal{E}_C$, this path is of even length. Due to the claim, it holds that $x\left(E(P)\right) = \frac{|P|}{2}$.

The first and the last edge in P are the edges e^* and f^* (in some order), and for both edges we have $x_{e^*} = x_{f^*} = 0$. Analogous to the equality (6.12) this implies that $x(E(P)) = \frac{|P|-2}{2}$, which contradicts $x(E(P)) = \frac{|P|}{2}$. □

One should note that in Lemma 6.6 it is also possible that $\mathcal{O}_C = \emptyset$ or $\mathcal{E}_C = \emptyset$. Now, we introduce a class of valid inequalities for $\operatorname{conv}(S)$. An example for the validity of the inequalities is depicted in Figure 6.2.

Theorem 6.7. *Let C be an unbalanced cycle in the graph G and let $\mathcal{P}_C = (\mathcal{O}_C, \mathcal{E}_C)$ be a parity on-level partition of C with $|\mathcal{O}_C| > |\mathcal{E}_C|$.*
Let $\nu := k - |\mathcal{O}_C| + 1$, where k is the parameter of the level constraint (6.8). Furthermore, let $\mathcal{T}_C := \Big\{(e_{i_1}, \dots, e_{i_\nu}) \mid e_{i_j} \in E_{on} \setminus C, e_{i_s} \neq e_{i_t} \text{ for all } s \neq t\Big\}$ be the set of ν-tuples of on-level edges in E which do not appear in the cycle C.
Then, the inequalities $(\mathcal{I}_C^1)$ and $(\mathcal{I}_C^2)$ as defined next are valid for $\operatorname{conv}(S)$.

i) If $k < |\mathcal{O}_C|$:

$$\sum_{e \in E(C)} x_e - x_{e'} \leq \frac{|C|}{2} - 1 \qquad \forall\ e' \in \delta(N(\mathcal{O}_C)) \cap C_{\mathit{off}} \qquad (\mathcal{I}_C^1)$$

ii) If $k \geq |\mathcal{O}_C|$:

$$\sum_{e \in E(C)} x_e - x_{e'} + \sum_{f \in T} x_f \leq \frac{|C|}{2} - 1 + \nu \qquad \forall\ e' \in \delta(N(\mathcal{O}_C)) \cap C_{\mathit{off}}, \forall\ T \in \mathcal{T}_C \qquad (\mathcal{I}_C^2)$$

Proof. We start with showing case i): Let x be any point in S. We assume that $x(E(C)) = \frac{|C|}{2}$, as it is clear that all the inequalities in $(\mathcal{I}_C^1)$ are satisfied by x if $x(E(C)) \leq \frac{|C|}{2} - 1$. Thus, due to Lemma 6.6, $x_e = 1$ either for all $e \in \mathcal{O}_C$ or for all $e \in \mathcal{E}_C$. As $k < |\mathcal{O}_C|$, it cannot hold that $x_e = 1$ for all $e \in \mathcal{O}_C$. Hence, $x_e = 0$ for all $e \in \mathcal{O}_C$. When $x(E(C)) = \frac{|C|}{2}$ it holds that two incident edges $e, f \in E(C)$ satisfy either $x_e = 1, x_f = 0$ or $x_e = 0, x_f = 1$. Therefore, $x_{e'} = 1$ and x satisfies the inequalities $(\mathcal{I}_C^1)$.

Now, we consider case ii): Again we assume that $x(E(C)) = \frac{|C|}{2}$, as the bounds $-x_{e'} \leq 0$ and $x(T) \leq \nu$ yield that all the inequalities in $(\mathcal{I}_C^2)$ are satisfied by x if $x(E(C)) \leq \frac{|C|}{2} - 1$. We use Lemma 6.6 and the fact that two incident edges $e, f \in E(C)$ satisfy either $x_e = 1, x_f = 0$ or $x_e = 0, x_f = 1$. If $x_e = 1$ for all $e \in \mathcal{E}_C$, then $x_{e'} = 1$ and x satisfies the inequalities $(\mathcal{I}_C^2)$.

If $x_e = 1$ for all $e \in \mathcal{O}_C$, then there are exactly $k - |\mathcal{O}_C|$ on-level edges f in $E_{\text{on}} \setminus C$ with $x_f = 1$. Hence,

$$\sum_{f \in T} x_f \leq k - |\mathcal{O}_C| = \nu - 1.$$

As a consequence, x satisfies the inequalities $(\mathcal{I}_C^2)$. □

Note that given an unbalanced cycle C in G and a parity on-level partition $\mathcal{P}_C$, exactly one of the types of inequalities $(\mathcal{I}_C^1)$ or $(\mathcal{I}_C^2)$ applies. We summarize both types of inequalities under the term $(\mathcal{I}_C)$.

Next, we consider the fractional vertices of the polytope P. For each such vertex $\bar{x}$ there exists an inequality in $(\mathcal{I}_C)$ which is violated by $\bar{x}$. Before we show this, we establish as an auxiliary result Lemma 6.8.

Lemma 6.8. *Let $\bar{x} \in P$ be a non-integral vertex of the polytope P. Then, there is a unique fractional cycle C in G. This cycle is unbalanced.*

Proof. As first part of this proof we show that all fractional edges in E, i.e. edges $e \in E$ with $0 < \bar{x}_e < 1$, build an unbalanced cycle in G. The point $\bar{x}$ contains at least one fractional component. As $\bar{x}$ fulfills constraints (6.6) and (6.7), there must be a cycle C in G such that $0 < \bar{x}_e < 1$ for all $e \in E(C)$. Furthermore, the cycle C is unbalanced regarding any parity on-level partition $(\mathcal{O}_C, \mathcal{E}_C)$. To see this, we assume that $|\mathcal{O}_C| = |\mathcal{E}_C|$. Let $C = [z_{i_1}, e_{i_1}, z_{i_2}, \dots, z_{i_p}, e_{i_p}, z_{i_1}]$ and let $\epsilon := \min\{\bar{x}_e, 1 - \bar{x}_e \mid e \in E(C)\}$. We define the points $\bar{x}'$ and $\bar{x}''$ by

$$\bar{x}'_e := \begin{cases} \bar{x}_e, & \text{if } e \notin E(C); \\ \bar{x}_e + \epsilon, & \text{if } e = e_{i_j} \in E(C) \text{ and } j \text{ is odd;} \\ \bar{x}_e - \epsilon, & \text{if } e = e_{i_j} \in E(C) \text{ and } j \text{ is even;} \end{cases}$$

$$\bar{x}''_e := \begin{cases} \bar{x}_e, & \text{if } e \notin E(C); \\ \bar{x}_e - \epsilon, & \text{if } e = e_{i_j} \in E(C) \text{ and } j \text{ is odd;} \\ \bar{x}_e + \epsilon, & \text{if } e = e_{i_j} \in E(C) \text{ and } j \text{ is even.} \end{cases}$$

As ϵ is added with alternating sign to the components of $\bar{x}$ which correspond to edges in C, the points $\bar{x}'$ and $\bar{x}''$ both satisfy constraints (6.6) and (6.7). As ϵ is chosen sufficiently small, $0 \le \bar{x}'_e \le 1$ and $0 \le \bar{x}''_e \le 1$ for all $e \in E$. Moreover, as C is assumed to be balanced, both points also satisfy constraint (6.8). Hence, $\bar{x}', \bar{x}'' \in P$. Now, $\bar{x}$ can be written as $\bar{x} = \frac{1}{2}(\bar{x}' + \bar{x}'')$, which contradicts $\bar{x}$ being a vertex of P.

So, we have proven that there is a fractional cycle C in G which is unbalanced. It remains to be shown that this cycle is unique. To this end, we assume that there exist 2 cycles C_1 and C_2 in G, with $0 < \bar{x}_e < 1$ for all $e \in E(C_1) \cup E(C_2)$ and with at least one edge $e \in E(C_1)$ with $e \notin E(C_2)$. As a balanced cycle contradicts $\bar{x}$ being a vertex of P, the cycles C_1 and C_2 both must be unbalanced. Let $(\mathcal{O}_{C_1}, \mathcal{E}_{C_1})$ and $(\mathcal{O}_{C_2}, \mathcal{E}_{C_2})$ be parity on-level partitions of C_1 and C_2, respectively.

Let $C_1 = [z_{i_1}, e_{i_1}, z_{i_2}, \dots, z_{i_p}, e_{i_p}, z_{i_1}]$ and let $C_2 = [z_{h_1}, e_{h_1}, z_{h_2}, \dots, z_{h_q}, e_{h_q}, z_{h_1}]$ with the property that an on-level edge $e_{i_j} \in E(C_1)$ is in $\mathcal{O}_{C_1}$ if and only if j is odd and an on-level edge $e_{h_l} \in E(C_2)$ is in $\mathcal{O}_{C_2}$ if and only if l is odd.

For an arbitrary $\epsilon_1 > 0$ let

$$\epsilon_2 := -\frac{|\mathcal{O}_{C_1}| - |\mathcal{E}_{C_1}|}{|\mathcal{O}_{C_2}| - |\mathcal{E}_{C_2}|}\epsilon_1.$$

We define an auxiliary point y' by alternatingly adding ϵ_1 and $-\epsilon_1$ to the components of $\bar{x}$ which correspond to the edges in C_1, starting with edge e_{i_1}. Then, the point $\bar{x}'$ is defined by alternatingly adding ϵ_2 and $-\epsilon_2$ to the components of y' which correspond to the edges in C_2, starting with edge e_{h_1}.

$$y'_e := \begin{cases} \bar{x}_e, & \text{if } e \notin E(C_1); \\ \bar{x}_e + \epsilon_1, & \text{if } e = e_{i_j} \in E(C_1) \text{ and } j \text{ is odd;} \\ \bar{x}_e - \epsilon_1, & \text{if } e = e_{i_j} \in E(C_1) \text{ and } j \text{ is even;} \end{cases}$$

$$\bar{x}'_e := \begin{cases} y'_e, & \text{if } e \notin E(C_2); \\ y'_e + \epsilon_2, & \text{if } e = e_{h_j} \in E(C_2) \text{ and } j \text{ is odd;} \\ y'_e - \epsilon_2, & \text{if } e = e_{h_j} \in E(C_2) \text{ and } j \text{ is even.} \end{cases}$$

Next, we define an auxiliary point y'' by alternatingly adding $-\epsilon_1$ and ϵ_1 to the components of $\bar{x}$ which correspond to the edges in C_1, starting with edge e_{i_1}. Then, the point $\bar{x}''$ is defined by alternatingly adding $-\epsilon_2$ and ϵ_2 to the components of y'' which correspond to the edges in C_2, starting with edge e_{h_1}.

$$y''_e := \begin{cases} \bar{x}_e, & \text{if } e \notin E(C_1); \\ \bar{x}_e - \epsilon_1, & \text{if } e = e_{i_j} \in E(C_1) \text{ and } j \text{ is odd;} \\ \bar{x}_e + \epsilon_1, & \text{if } e = e_{i_j} \in E(C_1) \text{ and } j \text{ is even;} \end{cases}$$

$$\bar{x}''_e := \begin{cases} y''_e, & \text{if } e \notin E(C_2); \\ y''_e - \epsilon_2, & \text{if } e = e_{h_j} \in E(C_2) \text{ and } j \text{ is odd;} \\ y''_e + \epsilon_2, & \text{if } e = e_{h_j} \in E(C_2) \text{ and } j \text{ is even.} \end{cases}$$

As ϵ_1 or ϵ_2 are added with alternating sign to cycle components of $\bar{x}$, both points $\bar{x}'$ and $\bar{x}''$ satisfy constraints (6.6) and (6.7). For ϵ_1 being sufficiently small, we have $0 \leq \bar{x}'_e \leq 1$ and $0 \leq \bar{x}''_e \leq 1$ for all $e \in E$. The difference in the sum of the on-level components of $\bar{x}'$ and the sum of the on-level components of $\bar{x}$ is

$$\bar{x}'(E_{\text{on}}) - \bar{x}(E_{\text{on}}) = |\mathcal{O}_{C_1}|\epsilon_1 - |\mathcal{E}_{C_1}|\epsilon_1 + |\mathcal{O}_{C_2}|\epsilon_2 - |\mathcal{E}_{C_2}|\epsilon_2.$$

Together with the definition of ϵ_2 this yields

$$\bar{x}'(E_{\text{on}}) - \bar{x}(E_{\text{on}}) = |\mathcal{O}_{C_1}|\epsilon_1 - |\mathcal{E}_{C_1}|\epsilon_1 - \frac{(|\mathcal{O}_{C_2}| - |\mathcal{E}_{C_2}|)(|\mathcal{O}_{C_1}| - |\mathcal{E}_{C_1}|)}{|\mathcal{O}_{C_2}| - |\mathcal{E}_{C_2}|}\epsilon_1 = 0,$$

and thus $\bar{x}'(E_{\text{on}}) = \bar{x}(E_{\text{on}}) = k$. Analogously,

$$\bar{x}''(E_{\text{on}}) - \bar{x}(E_{\text{on}}) = -|\mathcal{O}_{C_1}|\epsilon_1 + |\mathcal{E}_{C_1}|\epsilon_1 - |\mathcal{O}_{C_2}|\epsilon_2 + |\mathcal{E}_{C_2}|\epsilon_2 = 0,$$

and thus $\bar{x}''(E_{\text{on}}) = \bar{x}(E_{\text{on}}) = k$.

Hence, $\bar{x}'$ and $\bar{x}''$ both satisfy constraint (6.8) and thus both points are in P. Now, $\bar{x}$ can be written as $\bar{x} = \frac{1}{2}(\bar{x}' + \bar{x}'')$, which contradicts $\bar{x}$ being a vertex of P.

So, there is exactly one fractional cycle C in G. As we have shown in the beginning of the proof, this cycle additionally is unbalanced. $\square$

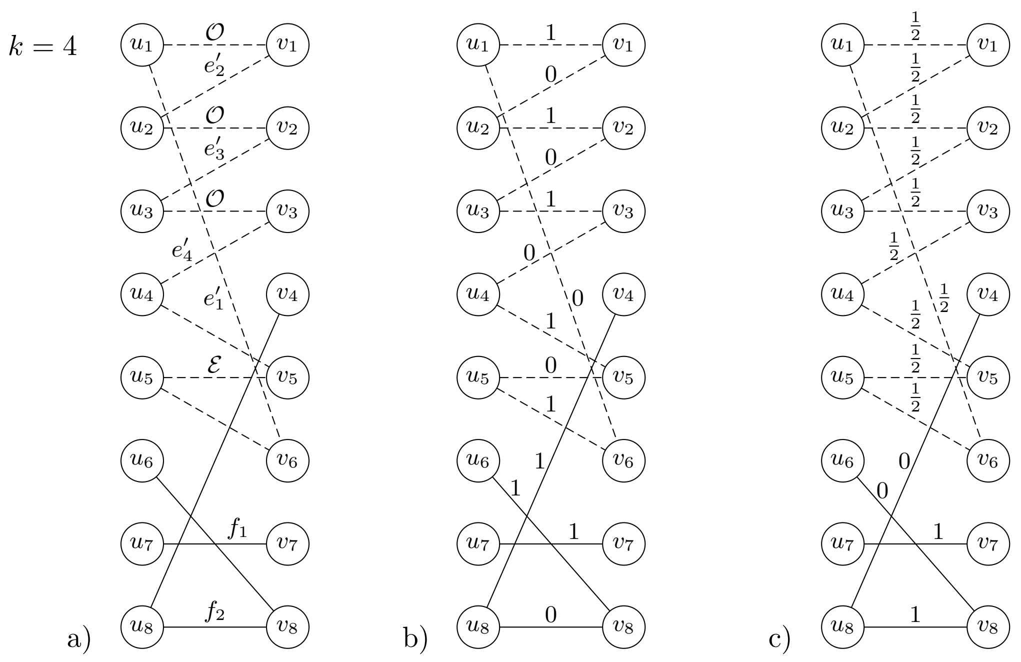


Figure 6.2.: a) Example of an LCPMP instance with an unbalanced cycle C indicated with dashed edges, and a parity on-level partition $(\mathcal{O}, \mathcal{E})$ of C with $|\mathcal{O}| = 3$ and $|\mathcal{E}| = 1$. In inequalities $(\mathcal{I}_C)$ we have that $e' \in \{e'_1, \ldots, e'_4\}$, $\nu = 2$ and $\mathcal{T}_C = \{(f_1, f_2)\}$. b) A point $x \in S$ with $x(E(C)) = \frac{|C|}{2}$, which satisfies $(\mathcal{I}_C^2)$. c) A fractional vertex $\bar{x} \in P$, which violates $(\mathcal{I}_C^2)$ for all $e' \in \{e'_1, \ldots, e'_4\}$.

Theorem 6.9. *Let $\bar{x} \in P$ be a non-integral vertex of the polytope P. Then, there is an inequality in $(\mathcal{I}_C)$ which is not satisfied by $\bar{x}$.*

Proof. Due to Lemma 6.8 there exists a unique fractional cycle C in the graph G, which in addition is unbalanced. Let $\mathcal{P}_C = (\mathcal{O}_C, \mathcal{E}_C)$ be a parity on-level partition of C with $|\mathcal{O}_C| > |\mathcal{E}_C|$ and let $\nu := k - |\mathcal{O}_C| + 1$.

As C is the only cycle in G with $0 < \bar{x}_e < 1$ for all $e \in E(C)$ and constraints (6.6) and (6.7) are satisfied for all $z \in N(C)$, it holds that $\bar{x}(E(C)) = \frac{|C|}{2}$. Furthermore, it holds that $\bar{x}_{e'} < 1$ for all $e' \in \delta(N(\mathcal{O}_C)) \cap C_{\text{off}}$ as e' is an edge in the cycle C.

i) If $k < |\mathcal{O}_C|$, the point $\bar{x}$ does not satisfy any inequality in $(\mathcal{I}_C^1)$. This follows directly from $\bar{x}(E(C)) = \frac{|C|}{2}$ and $\bar{x}_{e'} < 1$ for all $e' \in \delta(N(\mathcal{O}_C)) \cap C_{\text{off}}$.

ii) If $k \geq |\mathcal{O}_C|$, we make use of the fact that C contains all edges $e \in E$ where $\bar{x}_e$ is fractional. Thus, it holds that $\bar{x}_f \in \{0, 1\}$ for all $f \in E_{\text{on}} \setminus C_{\text{on}}$. Due to the fact that $\bar{x}(E_{\text{on}}) = k$, this further implies that $\bar{x}(C_{\text{on}})$ is integral. Together with the fact that $\bar{x}(C_{\text{on}}) < |\mathcal{O}_C|$, as we have stated in Lemma 6.4, it follows that $\bar{x}(C_{\text{on}}) \leq |\mathcal{O}_C| - 1$.

Hence, there is a tuple $T_0 \in \mathcal{T}_C$ with $\sum_{f \in T_0} \bar{x}_f \geq k - (|\mathcal{O}_C| - 1) = \nu$. So,

$$\sum_{e \in E(C)} \bar{x}_e - \bar{x}_{e'} + \sum_{f \in T_0} \bar{x}_f > \frac{|C|}{2} - 1 + \nu.$$

□

An example of a fractional vertex of P cut off by an inequality in $(\mathcal{I}_C^2)$ can be also found in Figure 6.2. The proof of Theorem 6.9 reveals how to determine an explicit inequality in $(\mathcal{I}_C)$ that cuts off an non-integral vertex from P.

Corollary 6.10. *Let $\bar{x} \in P$ be a non-integral vertex of the polytope P. Then, an inequality $\sum_{i=1}^{m} a_i x_i \leq a_0$ in $(\mathcal{I}_C)$ which is not satisfied by $\bar{x}$ can be found in $\mathcal{O}(m)$, where $m := |E|$.*

Proof. The fractional cycle C in G can be found in $\mathcal{O}(m)$ by first searching for an edge $e_{i_1} \in E$ with $0 < \bar{x}_{e_{i_1}} < 1$ and then iteratively searching for the fractional edge that shares an end-node with the current fractional edge. A parity on-level partition $(\mathcal{O}_C, \mathcal{E}_C)$ of C with $|\mathcal{O}_C| > |\mathcal{E}_C|$ can be determined by going through the edges of the cycle C in the appropriate order. While doing so, we can also determine an edge $e' \in \delta(N(\mathcal{O}_C)) \cap C_{\text{off}}$, with $\bar{x}_{e'} \leq \bar{x}_e$ for all $e \in \delta(N(\mathcal{O}_C)) \cap C_{\text{off}}$.

i) If $k < |\mathcal{O}_C|$, then we have already determined all components of a violated inequality in $(\mathcal{I}_C^1)$.

ii) If $k \geq |\mathcal{O}_C|$, then we further need to determine a tuple $T_0 \in \mathcal{T}_C$ such that $T_0 \in \operatorname{argmax}_{T \in \mathcal{T}_C} \bar{x}(T)$. By checking (at most) all components of $\bar{x}$ which belong to on-level edges in $E \setminus C$, this can be done in $\mathcal{O}(m)$. Then, all components of a violated inequality in $(\mathcal{I}_C^2)$ are determined.

In total, all steps can be performed in $\mathcal{O}(m)$. □

We conclude this section by mentioning that the total number of inequalities in $(\mathcal{I}_C)$ needed to cut off all non-integral vertices of P might be exponential in n, as for each cycle C in G there exists a set of inequalities based on this cycle. Nevertheless, we have shown how to determine a separating inequality for a specific non-integral vertex in polynomial time.

7. Novel Solution Algorithms for Equality Constrained Matching

We have seen in Section 4.3.2 that perfect matchings satisfying one additional equality side constraint in a complete bipartite graph can be found using a polynomial algorithm by Yi, Murty and Spera [52]. It remains an open question whether there is a polynomial algorithm for this problem on general bipartite graphs, which are not necessarily complete. In this chapter we present novel solution approaches for this problem with emphasis on the level constraint as side constraint.

For the level constrained matching problem on level graphs we develop a polynomial approximation algorithm in Section 7.1. A matching returned by this algorithm satisfies the level constraint and is of size at least $z^* - 1$, where z^* is the size of an optimal solution.

In Section 7.2 we develop an exact solution algorithm for the equality constrained perfect matching problem on bipartite graphs. We describe the design of the corresponding algorithm and give notes on its implementation.

7.1. Approximation algorithm for the level constrained matching problem

The aim of this section is to develop a polynomial approximation algorithm for the LCMP on level graphs. The LCMP-approximation algorithm guarantees to return matchings which satisfy the level constraint and which have a cardinality that differs from the cardinality of an optimal solution by at most 1. Furthermore, if the optimal solution of an LCMP instance is not a *perfect* matching, the LCMP-approximation algorithm even returns an optimal solution. The importance of this algorithm lies in the fact that no polynomial solution algorithm for the LCMP is known – even when it is restricted to level graphs (compare Section 4.1).

7.1.1. The LCMP-approximation algorithm

The approximation procedure can be broken down into two phases. First, it determines the largest matching which contains *at most* as many on-level edges as the parameter of the level constraint. Most likely, this matching does not satisfy the level constraint with equality, otherwise an optimal solution has been found. Second, the matching is modified to satisfy the level constraint by adding the missing number of on-level edges to it. Of course, this step entails that new edges are added and some existing edges are removed from the formerly found matching.

Algorithm 7.1 Approximation algorithm for level constrained matching

```
LCMP-Approx($G, k$)
 Input: Bipartite level graph $G = (U \uplus V, E)$ with $U = \{u_1, \ldots, u_n\}, V = \{v_1, \ldots, v_n\}$,
   integer $k$ with $0 \leq k \leq n$
 Output: Matching $M$ in $G$ with $|M \cap E_{\text{on}}| = k$
 1: Phase 1:
 2: $M \leftarrow$ maximum matching $\bar{M}$ in $G$ s.t. $|\bar{M} \cap E_{\text{on}}| \leq k$      ▷ Minimum cost flow problem
 3: if $M \cap E_{\text{on}} = k$ then
 4:     return $M$
 5: Phase 2:
 6: $z \leftarrow k - |M \cap E_{\text{on}}|$                                   ▷ Level constraint deficit
 7: Construct graph $S_M = (W, F)$ with
    $W \leftarrow \{w_i \mid i = 1, \ldots, n : [u_i, v_i] \notin M\}$,
    $F \leftarrow \{[w_i, w_j]^i \mid [u_i, v_j] \in M, u_i \in U, v_j \in V, i \neq j\}$
 8: $M' \leftarrow M$; $z' \leftarrow z$
 9: if there exists a node $w_i \in W$ with $\deg_{S_M}(w_i) = 1$ then                ▷ Path case
10:     insert($[u_i, v_i], M'$)                         ▷ Insertion corrsp. to first node of path
11:     $z' \leftarrow z' - 1$; mark node $w_i$
12:     while $z' > 0$ and $w_i$ has an unmarked neighbor node $x$ in $S_M$ do
13:         $i \leftarrow j \in \{1, \ldots, n\}$ such that $w_j = x$
14:         insert($[u_i, v_i], M'$)                    ▷ Insertion corrsp. to other nodes in path
15:         $z' \leftarrow z' - 1$; mark node $w_i$
16: if $z' = 0$ then
17:     return $M'$
18: while $z' > 0$ and there exists an unmarked node $w_i$ in $S_M$ do              ▷ Cycle case
19:     insert($[u_i, v_i], M'$)              ▷ Insertion corresp. to first node $w_i$ of cycle $C_i$
20:     $z' \leftarrow z' - 1$; mark node $w_i$
21:     while $z' > 0$ and $w_i$ has an unmarked neighbor node $x$ in $S_M$ do
22:         $i \leftarrow j \in \{1, \ldots, n\}$ such that $w_j = x$
23:         insert($[u_i, v_i], M'$)            ▷ Insertion corresp. to other nodes in cycle $C_i$
24:         $z' \leftarrow z' - 1$; mark node $w_i$
25: return $M'$

26: procedure insert($[u_i, v_i], M$)
27:     while $M$ contains an edge $e$ incident to $u_i$ or $v_i$ do
28:         $M \leftarrow M \setminus \{e\}$
29:     $M \leftarrow M \cup \{[u_i, v_i]\}$
```

The LCMP-approximation algorithm is listed as Algorithm 7.1. In the following paragraphs we describe the individual steps of the algorithm. Its correctness, approximation quality and its running-time are proven in Theorem 7.4.

In order to describe how the two phases of the approximation procedure work exactly, we consider the algorithm being applied to a problem instance $(G = (U \uplus V, E), k)$ of the LCMP. In this instance, G is a bipartite level graph with node sets $U = \{u_1, \dots, u_n\}$ and $V = \{v_1, \dots, v_n\}$. As G is a level graph, it contains an edge $[u_i, v_i]$ for all $i = 1, \dots, n$. Further, k is a nonnegative integer with $k \leq n$. Let $m := |E|$ denote the number of edges in G.

Phase 1

The objective of *Phase 1* of the LCMP-approximation algorithm is to find a matching of maximum size in G which satisfies a weakened form of the level constraint. The matching is not needed to contain exactly k on-level edges, it rather is required to contain at most k of them. More precisely, the task is to determine an optimal solution to the following problem:

$$\begin{aligned} \max \quad & \sum_{e \in E} x_e && && (7.1)\\ \text{s.t.} \quad & \sum_{e \in \delta(u_i)} x_e \leq 1 && \forall\, i = 1, \dots, n && (7.2)\\ & \sum_{e \in \delta(v_i)} x_e \leq 1 && \forall\, i = 1, \dots, n && (7.3)\\ & \sum_{i=1}^{n} x_{[u_i, v_i]} \leq k && && (7.4)\\ & x_e \in \{0, 1\} && \forall\, e \in E. && (7.5) \end{aligned}$$

To find an optimal solution M of this problem, a series of minimum cost flow problem instances is solved. Each single minimum cost flow problem instance is used to find – if it exists – a matching M_r in the graph G which is of a specific size r and which satisfies the upper bound constraint (7.4). Then, the matching with biggest cardinality among $M_1, \dots, M_n$ is a solution to problem (7.1) – (7.5).

We now define the instance of the minimum cost flow problem (see Section 1.1.2) which is used to calculate a matching M_r for some $r = 1, \dots, n$. Let $D = (N, A)$ be a digraph with a source node $s \in N$ and a sink node $t \in N$. For each arc $a \in A$, its capacity is denoted by p_a and its cost is denoted by c_a. Further, the value b_z is the supply/demand of a node $z \in N$. The digraph D and the parameters of the minimum cost flow problem instance are defined as follows:

$$\begin{aligned}
&N := U \cup V \cup \{s,t\},\\
&A := \{(s,u_i) \mid u_i \in U\} \cup \{(v_j,t) \mid v_j \in V\} \cup \{(u_i,v_j) \mid [u_i,v_j] \in E \text{ with } u_i \in U, v_j \in V\},\\
&p_a := 1 \quad \text{for all } a \in A,\\
&c_{(u,v)} := \begin{cases} 1 & \text{if there is an } i \in \{1,\dots,n\} \text{ such that } u = u_i \in U \text{ and } v = v_i \in V,\\ 0 & \text{else,} \end{cases}\\
&b_z := \begin{cases} r & \text{if } z = s,\\ -r & \text{if } z = t,\\ 0 & \text{else.} \end{cases}
\end{aligned}$$

An optimal b-flow in the digraph D corresponds to a matching in the graph G which is of size r and contains a minimum number of on-level edges.

Let M_i denote the matching corresponding to an optimal solution of the above defined minimum cost flow problem with the parameter $r = i$. Let M be that matching among $M_1,\dots,M_n$ which has maximum cardinality and satisfies the upper bound constraint (7.4). It can be found efficiently among $M_1,\dots,M_n$ by a binary search approach. This avoids determining M_i for all $i = 1,\dots,n$, which would imply solving the minimum cost flow problem for all parameter values $r = 1,\dots,n$ (though even that could be done in polynomial time).

If M satisfies the upper bound constraint with equality, it fulfills the level constraint from the initial LCMP instance. In this case, the LCMP-approximation algorithm returns the matching M as a solution. If M does not satisfy the upper bound constraint with equality, the LCMP-approximation algorithm continues with Phase 2.

Phase 2

The objective of *Phase 2* is to modify the matching M from Phase 1 such that it contains exactly k on-level edges and hence satisfies the level constraint. In Figure 7.1 the main steps of Phase 2 are illustrated at an example.

The difference between the demanded number of on-level edges and the number of on-level edges in M is denoted by z, i.e.

$$z = k - |\{[u_i,v_i] \in M \mid i = 1,\dots,n\}|.$$

Whenever an on-level edge $[u_i,v_i]$ is added to M, this is done according to the following definition:

Definition 7.1. *Let $G = (U \mathbin{\dot\cup} V, E)$ be a bipartite level graph with $U = \{u_1,\dots,u_n\}$ and $V = \{v_1,\dots,v_n\}$. Let M be a matching in G and let $e = [u_i,v_i]$ be an on-level edge in G with $e \notin M$.*

When we say that the on-level edge e is inserted *into M, this implies that the following steps are done:*

1. *Removal of those edges from M which are incident to either u_i or v_i,*
2. *Adding $[u_i, v_i]$ to M.*

In Algorithm 7.1, the procedure INSERT carries out the insertion of on-level edges into matchings.

In order to determine which on-level edges are to be inserted into M, we define an auxiliary graph S_M. This graph results from restricting G to contain only those edges which are in M, and shrinking it by identifying the two nodes u_i and v_i for all $i = 1, \ldots, n$. One should note that the resulting graph may contain loops resulting from the on-level edges which are in M already, and it may contain parallel edges due to symmetric edges in M. We remove all loops including their associated nodes. Parallel edges are kept and are distinguished by a superscript.

The graph $S_M = (W, F)$ is defined by

$$\begin{aligned} W &= \{w_i \mid i = 1, \ldots, n : [u_i, v_i] \notin M\}, \\ F &= \left\{[w_i, w_j]^i \mid [u_i, v_j] \in M, u_i \in U, v_j \in V, i \neq j\right\}. \end{aligned}$$

The superscript i of an edge in F denotes that the edge is induced by an edge in M with end-node $u_i \in U$. This enables S_M to contain parallel edges $[w_i, w_j]^i, [w_i, w_j]^j$ which are induced by two edges of the form $[u_i, v_j], [u_j, v_i]$ in M, with $i \neq j$.

As the maximum node degree in S_M is 2, there is a unique partition of S_M into node-disjoint paths and cycles. Let $\mathcal{P}$ and $\mathcal{C}$ denote the set of these paths and cycles, respectively. The first step of increasing the number of on-level edges in M depends on whether there exists a path in this partition or not. This can easily be figured out by checking whether there is a node in S_M which has degree 1.

If there is a node $w_{i_1} \in W$ with $\deg_{S_M}(w_{i_1}) = 1$, then let $P = [w_{i_1}, w_{i_2}, \ldots, w_{i_p}]$ be the unique path in $\mathcal{P}$ which contains the node w_{i_1}. If there is no such node, the following case discrimination is skipped and we define $M' := M$ and $z' := z$.

Case 1: $z < p$. The on-level edges $[u_{i_1}, v_{i_1}], \ldots, [u_{i_z}, v_{i_z}]$ are inserted into M. Let M_{app} be the resulting matching. The LCMP-approximation algorithm terminates and returns M_{app}.

Case 2: $z \geq p$. The on-level edges $[u_{i_1}, v_{i_1}], \ldots, [u_{i_p}, v_{i_p}]$ are inserted into M. Let M' be the resulting matching and let $z' := z - p$.

Now, the remaining deficit z' in the number of on-level edges in M' is balanced out. Let $C_1, \ldots, C_l$ be cycles in $\mathcal{C}$ such that their total length is at least z' and such that the total length of the cycles $C_1, \ldots, C_{l-1}$ is strictly less than z'. Such an l exists as each pair of nodes (u_i, v_i) in G either is already covered by an on-level edge in M' or there is a node w_i which represents this pair and which is in a cycle in S_M. Further, the total number of these nodes in S_M is $n - |M'| \geq k - |M'| = z'$.

Let q_i denote the length of the cycle C_i for all $i = 1, \ldots, l$. Further, let each cycle C_i be of the form $C_i = [w_{i_1}, \ldots, w_{i_{q_i}}, w_{i_1}]$. Then, the on-level edges $[u_{i_j}, v_{i_j}]$ are inserted into

M' for all $i = 1, \dots, l-1$ and $j = 1, \dots, q_i$. Let M'' denote the resulting matching and let $z'' := z' - \sum_{i=1}^{l-1} q_i$. Finally, the on-level edges $[u_{l_1}, v_{l_1}], \dots, [u_{l_{z''}}, v_{l_{z''}}]$ are inserted into M''. Let M_{app} be the resulting matching. The LCMP-approximation algorithm terminates and returns the matching M_{app}.

As far as the practical implementation is concerned, the cycles $C_1, \dots, C_l$ are not needed to be determined explicitly. We rather start with one cycle and successively insert on-level edges into the current matching with respect to the indices of the nodes in the cycle and the order in which the nodes appear in the cycle. We then consider the next cycle and continue this procedure, until the current matching contains exactly k on-level edges.

In order to find corresponding cycles in S_M, we proceed as follows. We keep a separate list of all nodes in S_M. In this list, all nodes appearing in P are marked. Whenever an on-level edge $[u_i, v_i]$ is added to the current matching, the corresponding node w_i in a cycle in S_M is marked as well. We go through this list in increasing order of the nodes' indices. When an unmarked node of degree 2 has been found, this node lies in a yet unconsidered cycle. The next time we need to find a cycle, we begin the search from the last node we have marked.

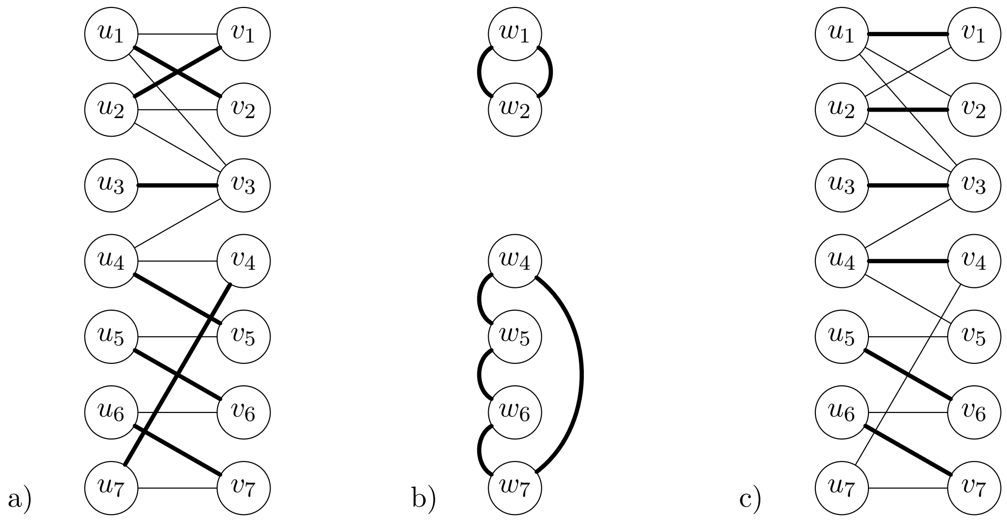


Figure 7.1.: a) Example of an instance of problem (7.1) – (7.5) with $k = 4$. An optimal solution matching M with at most k on-level edges is indicated with bold edges. b) Graph S_M, based on the matching M. The graph S_M can be partitioned into the two cycles $[w_1, w_2, w_1]$ and $[w_4, w_5, w_6, w_7, w_4]$. c) Bold edges indicate the matching M_{app} as returned by the LCMP-approximation algorithm. Here, M_{app} is an optimal solution to the LCMP instance.

Lemma 7.2. *Let $(G = (U \cup V, E), k)$ be a problem instance of the LCMP, where G is a bipartite level graph, $U = \{u_1, \dots, u_n\}, V = \{v_1, \dots, v_n\}$, $m = |E|$ and k is a nonnegative integer with $k \leq n$.*

Then, at the end of Phase 1 of the LCMP-approximation algorithm, the set M is an optimal solution to problem (7.1) – (7.5). The running-time of Phase 1 is $\mathcal{O}\left(n^2 m (\log n)^2\right)$.

Proof. In Phase 1, the problem to solve is a maximum matching problem with an additional upper bound side constraint. In Section 4.2 we presented a polynomial solution procedure for the perfect matching problem with a single upper bound side constraint, proposed by Itai et al. [29]. We slightly extend their approach in order to determine an optimal solution M to problem (7.1) – (7.5).

In the LCMP-approximation algorithm, minimum cost flow problems on the digraph D with different supply/demand values are solved. This is motivated by the equivalence of integral b-flows in D and matchings in G which are of size $r = b(s)$. The graph G contains a matching of size r which satisfies the upper bound constraint (7.4) if and only if the minimum cost flow problem with parameters $b(s) = r, b(t) = -r$ has an optimal solution f with $\sum_{i=1,\dots,n} f_{(u_i,v_i)} \leq k$.

Let M^* be an optimal solution to problem (7.1) – (7.5). The size of M^* ranges from 1 to n. Therefore, the biggest matching among $M_1, \dots, M_n$ which also satisfies the upper bound constraint is an optimal solution to problem (7.1) – (7.5).

Now, we calculate the running-time of Phase 1 of the LCMP-approximation algorithm. The construction of an instance of the minimum cost flow problem is linear in the size of the graph G. The digraph D contains $|N| = 2n + 2$ nodes and $|A| = m + 2n$ arcs. Additionally, all arcs are assigned a cost and a capacity value and all nodes are assigned a supply/demand value. After the first instance is created, the underlying digraph D, the arc costs and the arc capacities remain unchanged. The only two values that change in the different instances are the supply of node s and the demand of node t. Therefore, the construction of all n minimum cost flow problem instances is linear in the size of G.

For solving an instance of the minimum cost flow problem we consider the successive-shortest-path problem. Its running-time is $\mathcal{O}(|N| U S(D))$, where U is the maximum supply of any node in D and $S(D)$ denotes the time taken to solve a shortest path problem on the digraph D.

In each problem instance, the source node s has a supply of value $b(s) = r$, which is bounded above by n. Hence, each problem instance of the minimum cost flow problem can be solved in $\mathcal{O}\left(n^2 S(D)\right)$. We select Dijkstra's algorithm as the shortest path algorithm. It can be implemented on a heap structure such that its running-time on D is $\mathcal{O}\left(|A| \log |N|\right) = \mathcal{O}\left((m + n) \log n\right)$. Using Dijkstra's algorithm in the successive-shortest path algorithm yields a running-time of $\mathcal{O}\left((n^2 m + n^3) \log n\right)$. Considering that in a level graph G it holds that $m \geq \frac{1}{2}n$, each problem instance of the minimum cost flow problem can be solved in $\mathcal{O}\left(n^2 m \log n\right)$.

The matching M can be determined among $M_1, \dots, M_n$ by a binary search procedure (see [15]). We need to search for the biggest $i = 1, \dots, n$ such that M_i exists and contains at most k on-level edges. We can use a binary search approach, as the series of matchings $M_1, \dots, M_n$ has the following property: If M_i does not exist or does not satisfy the upper bound constraint (7.4), then M_j does not exist or satisfy the upper bound constraint for any $j \geq i$. This holds as all matchings of size j which are feasible for problem (7.2) – (7.5) contain a feasible matching of size i for all $i \leq j$.

In the course of the binary search procedure, a minimum cost flow problem instance with parameter $r = i$ is solved only when the key value i of the corresponding matching M_i is considered. Hence, the number of minimum cost flow problems which need to be solved to find the biggest matching M_i which satisfies the upper bound constraint (7.4) is in $\mathcal{O}(\log n)$.

This implies that Phase 1 has a total running-time of $\mathcal{O}\left(n^2 m (\log n)^2\right)$. $\square$

Lemma 7.3. *Let G, k be as in Lemma 7.2. Further, let M be the matching resulting from Phase 1 of the LCMP-approximation algorithm.*

Then, at the end of Phase 2 of the algorithm, the resulting set M_{app} is a matching in the graph G and satisfies the level constraint. The running-time of Phase 2 is $\mathcal{O}(n)$.

Proof. First, we show that at any point in Phase 2, the sets M, M' and M'' are matchings in the graph G and each of them contains at most k on-level edges.

We have seen in Lemma 7.2 that the set M resulting from Phase 1 is a matching and contains at most k on-level edges. All edges that are added to M in the course of Phase 2 are on-level edges. Before an edge of the form $[u_i, v_i]$ is added to the current matching, we remove those edges from it which are incident to either u_i or v_i. This ensures that after the on-level edge $[u_i, v_i]$ is added to M (M', M''), still no two edges in M (M', M'') meet at the same node.

The decision as to which on-level edges are to be added to the current matching depends on the graph S_M. The index of the end-nodes of an added on-level edge always coincides with the index of a node in S_M. As, by definition, the graph S_M does not contain a node w_i if the on-level edge $[u_i, v_i]$ is in M, all edges added to M (M', M'') in Phase 2 have not been in M (M', M'') before. Further, no on-level edges are removed from M (M', M'') in Phase 2. Hence, inserting a total number of exactly z on-level edges into M ensures that the number of on-level edges in the matching never exceeds k. Further, it finally yields a matching M_{app} which contains exactly k on-level edges and hence fulfills the level constraint.

Now, we calculate the running-time of Phase 2. Phase 2 starts with the construction of the graph S_M. By going through the edges in the matching M from Phase 1, the graph S_M can be constructed in the form of an adjacency list in $\mathcal{O}(n)$.

Let us now consider an on-level edge $[u_i, v_i]$ which is meant to be inserted into the current matching. The edges which are to be removed from the current matching can be found by first determining the edges in S_M which are incident to w_i. They can be determined in constant time, as S_M is stored in an adjacency list and each node has at most 2 incident edges. These edges are either $[w_i, w_j]^i$, $[w_i, w_j]^j$ or both, for some $j = 1, \dots, n$. They correspond to either $[u_i, v_j]$, $[u_j, v_i]$ or both edges in the current matching and can also be determined in constant time if stored appropriately. Further, adding an on-level edge to the current matching can be done in constant time. Thus, the entire action of inserting one on-level edge into the current matching can be done in constant time.

In total, $z \leq n$ on-level edges are inserted into M. The LCMP-approximation algorithm uses the indices of the nodes in the path P, if it exists, and in the cycles

$C_i, i = 1, \ldots, l$, to determine which on-level edges are inserted into M. Searching for a path P and the cycles C_i in S_M in the way described in the algorithm can be done in $\mathcal{O}(n)$. Hence, Phase 2 of the LCMP-approximation algorithm is completed in $\mathcal{O}(n)$. □

Theorem 7.4. *Let $(G = (U \cup V, E), k)$ be an instance of the LCMP, where G is a bipartite level graph, $U = \{u_1, \ldots, u_n\}, V = \{v_1, \ldots, v_n\}$, $m := |E|$ and k is a nonnegative integer with $k \leq n$. Let M_{opt} be an optimal solution to this problem instance.*

Then, the LCMP-approximation algorithm applied to this problem instance returns a matching M_{app} in G which satisfies the level constraint. It holds that $|M_{app}| \geq |M_{opt}| - 1$. In the case that M_{opt} is not a perfect matching, it holds that $|M_{app}| = |M_{opt}|$. The LCMP-approximation algorithm runs in $\mathcal{O}\left(n^2 m (\log n)^2\right)$.

Proof. It follows directly from Lemma 7.2 and Lemma 7.3 that M_{app} is a matching in the graph G which satisfies the level constraint. The running-time of the LCMP-approximation algorithm also is a direct consequence of these lemmata. Next, we show that the claimed approximation quality holds.

We have shown in Lemma 7.2 that at the end of Phase 1, the matching M is an optimal solution to the problem (7.1) – (7.5). As M_{opt} also is a feasible solution to this problem, it holds that $|M| \geq |M_{\text{opt}}|$.

Now, we investigate the changes in the cardinality of M during Phase 2. We start by considering the structure of the graph S_M. Due to its construction, the maximum node degree in S_M is 2. Further, if there would be a node w_i with $\deg(w_i) = 0$, both the nodes u_i and v_i in the graph G would not be covered by M. Then, the on-level edge $[u_i, v_i]$ could be added to M while still satisfying the upper bound constraint (7.4), as Phase 2 is only started when M contains strictly less than k on-level edges. This contradicts the fact that at the beginning of Phase 2, M is the biggest matching in G satisfying this upper bound constraint. Hence, each node w_i in S_M has $\deg(w_i) \in \{1, 2\}$. This implies that there is a partition of the graph S_M into pairwise node-disjoint paths and cycles. We denote these sets of paths and cycles by $\mathcal{P}$ and $\mathcal{C}$, respectively.

Whenever an on-level edge is inserted into M, there is a node w_i in the graph S_M, whose index determines the index of the end-nodes u_i and v_i of the added on-level edge. As all nodes w_i in S_M appear either in paths or in cycles of the sets $\mathcal{P}$ and $\mathcal{C}$, we investigate the following two cases:

- Let $P = [w_{i_1}, \ldots, w_{i_p}]$ be a path in $\mathcal{P}$. Let T be the set of nodes in P whose indices correspond to the indices of end-nodes of on-level edges which are added during Phase 2. According to the LCMP-approximation algorithm, T is always of the form $T = \{w_{i_1}, \ldots, w_{i_h}\}$, with $h \leq p$.

  If $h < p$, then there are h edges in P which are incident to nodes in T. Concerning the construction of S_M, these edges correspond to h edges in M. Therefore, when inserting the on-level edges $[u_{i_1}, v_{i_1}], \ldots, [u_{i_h}, v_{i_h}]$ into M, there are h edges which must be removed from M to ensure that M stays a matching. As a result, there is no difference in the size of M before and after inserting these on-level edges. A special case arises when $h = p$. When inserting the on-level edges $[u_{i_1}, v_{i_1}], \ldots, [u_{i_p}, v_{i_p}]$

into M, there are $p-1$ edges which must be removed from M to ensure that M stays a matching. As a result, after inserting these on-level edges into M, the size of M is increased by 1.

- Let $C_i = \left[w_{i_1}, \ldots, w_{i_{q_i}}, w_{i_1}\right]$ be a cycle in $\mathcal{C}$. Let T be the set of nodes in C_i whose indices correspond to the indices of end-nodes of on-level edges which are added during Phase 2. As all nodes in T appear consecutively in C_i, the set T can be written in the form $T = \{w_{i_1}, \ldots, w_{i_h}\}$, with $h \leq q_i$.

  If $h < q_i$, then there are $h+1$ edges in C_i which are incident to nodes in T. Therefore, when inserting the on-level edges $[u_{i_1}, v_{i_1}], \ldots, [u_{i_h}, v_{i_h}]$ into M, there are $h+1$ edges which must be removed from M to ensure that M stays a matching. As a result, the size of M decreases by 1. Also here a special case arises when $h = q_i$. When inserting the on-level edges $[u_{i_1}, v_{i_1}], \ldots, [u_{i_{q_i}}, v_{i_{q_i}}]$ into M, there are q_i edges which must be removed from M to ensure that M stays a matching. As a result, there is no difference in the size of M before and after inserting these on-level edges.

Now, we consider the case that at the beginning of Phase 2 there exists a path $P = [w_{i_1}, \ldots, w_{i_p}]$ in $\mathcal{P}$. This case occurs if M is not a perfect matching in G, as this implies at least one unmatched node u_i or v_i in G, which results in a node w_i in S_M with degree $\deg(w_i) = 1$.

If $z < p$, inserting on-level edges into M as described in the LCMP-approximation algorithm yields a matching M_{app} with

$$|M_{\text{app}}| = |M| \geq |M_{\text{opt}}|.$$

If $z \geq p$, first all on-level edges $[u_{i_1}, v_{i_1}], \ldots, [u_{i_p}, v_{i_p}]$ are inserted into M, which yields a matching M' with $|M'| = |M| + 1$. The case $z = p$ is not possible, as otherwise M' would be a matching which satisfies the level constraint and which is of bigger size than an optimal solution. Hence, there still must be a deficit in the number of on-level edges in M' and the LCMP-approximation algorithm proceeds as follows.

First, for the cycles $C_1, \ldots, C_{l-1}$, on-level edges corresponding to *all* nodes in these cycles are inserted into M'. This yields the matching M'' with $|M''| = |M'|$. Second, on-level edges corresponding to a proper subset of nodes in a cycle C_l are inserted into M''. This results in a matching M_{app} with

$$|M_{\text{app}}| = |M''| - 1 = |M'| - 1 = |M| \geq |M_{\text{opt}}|. \tag{7.6}$$

One should note that the number of cycles in $\mathcal{C}$ is always sufficient to add enough on-level edges this way. To see this, we recall that each node in S_M stands for the possibility to add an on-level edge to M, and all nodes in S_M appear either in a path in $\mathcal{P}$ or in a cycle in $\mathcal{C}$. Furthermore, if $z > p$, then P is the only path in $\mathcal{P}$, as otherwise one could insert on-level edges such that the resulting matching satisfies the level constraint and its size would be greater than $|M_{\text{opt}}|$.

Now, we consider the case that $\mathcal{P}$ is empty. Then, we have that $M' = M$. This changes the equality (7.6) as follows:

$$|M_{\text{app}}| = |M''| - 1 = |M'| - 1 = |M| - 1 \geq |M_{\text{opt}}| - 1.$$

In any case, the LCMP-approximation algorithm returns a matching M_{app} which satisfies the level constraint and which is of size $|M_{\text{app}}| \geq |M_{\text{opt}}| - 1$. □

Theorem 7.4 states that the LCMP-approximation algorithm returns a solution matching M_{app} whose size is at least $|M_{\text{opt}}| - 1$, where M_{opt} is an optimal solution to the underlying LCMP instance. As the graph in the LCMP instance is a level graph, we can assume that $|M_{\text{opt}}| \geq 1$. Then, the following estimation holds:

$$\frac{|M_{\text{app}}|}{|M_{\text{opt}}|} \geq 1 - \frac{1}{|M_{\text{opt}}|}. \tag{7.7}$$

Let $(G = (U \mathbin{\dot\cup} V, E), k))$ be a LCMP instance, where G is a bipartite level graph with $|U| = |V| = n$ and k is a nonnegative integer with $k \leq n$. The only case where M_{app} may be strictly smaller than M_{opt} is when M_{opt} is a perfect matching in G, i.e. when $|M_{\text{opt}}| = n$. So, when M_{opt} is not a perfect matching, estimation (7.7) simplifies to

$$\frac{|M_{\text{app}}|}{|M_{\text{opt}}|} = 1.$$

When M_{opt} is a perfect matching, the estimation (7.7) can be written as

$$\frac{|M_{\text{app}}|}{|M_{\text{opt}}|} \geq 1 - \frac{1}{n}.$$

Hence, the approximation ratio of our LCMP-approximation algorithm is $1 - \frac{1}{n}$, where n is the number of nodes in one color class of the graph G. This ratio is independent from the parameter k and converges to 1 with increasing number of nodes in G.

7.1.2. Improvement of the LCMP-approximation algorithm

It is possible to increase the number of problem instances for which the result of the LCMP-approximation algorithm is an optimal solution. To achieve this, we extend the part of the algorithm where cycles in $\mathcal{C}$ are considered to add the missing number of on-level edges z' to M'. More specifically, we enhance the selection of cycles $C_1, \ldots, C_l$ in the case that there is no path in $\mathcal{P}$.

In the LCMP-approximation algorithm, these cycles have the property that their total length is at least z' and that their total length without cycle C_l is strictly less than z'. The aim of the improvement is to choose these cycles such that their total length is exactly z', if possible. If such cycles exist, then on-level edges $[u_i, v_i]$ are inserted into M' for all nodes w_i in these cycles, and the number of edges removed from M' to preserve its matching property also is exactly z'. As a result, the number of edges added to M'

equals the number of edges removed from it. This is beneficial if the underlying LCMP instance has an optimal solution M_{opt} which is a perfect matching. In this case, a cycle C_l for which not all of its nodes are considered when inserting on-level edges causes a difference in the size of M_{app} and the size of M_{opt}. Using the above improvement, even if the underlying LCMP instance has a perfect matching as an optimal solution, the LCMP-approximation algorithm will return an optimal solution – provided that there exist cycles in $\mathcal{C}$ of total length equal to z'.

The problem of selecting cycles $C_1, \dots, C_l$ in $\mathcal{C}$ such that the sum of their lengths equals z' is the subset-sum problem. In general, the subset-sum problem is NP-complete (see [15]). Nevertheless, it can be solved pseudo-polynomially using a dynamic programming approach (see [15]):

Let $(v_1, \dots, v_a, y)$ be an instance of the subset-sum problem, with all parameters in this instance being nonnegative integer numbers (see Problem Formulation C.11 in Appendix C). The question is whether there is a subset of the values $v_1, \dots, v_a$ whose sum is equal to y. The dynamic programming approach to this problem is based on the idea of solving the instances $(v_1, \dots, v_j, s)$ for all $j = 1, \dots, a$ and $s = 1, \dots, y$. This can be done recursively, starting with the trivial problems where $j = 1$. Each recursive calculation can be done in constant time. Hence, solving all of the above instances, including the original problem where $j = a$ and $s = y$, can be done in $\mathcal{O}(ay)$.

Concerning the above problem of finding appropriate cycles, a is the number of cycles in G, which is bounded by n. The parameter y corresponds to z', which is also bounded by n, as the maximum number of missing on-level edges in M' is bounded by n. Thus, the dynamic programming approach applied to find a subset of cycles in $\mathcal{C}$ which have total length z' is polynomial in the input size of the graph G. Its running-time is in $\mathcal{O}(n^2)$. Using this improvement increases the running-time of Phase 2 but, the total running-time of the LCMP-approximation algorithm does not change.

It is clear that a subset of cycles in $\mathcal{C}$ with the desired property does not always exist. It depends on the structure of the matching M from Phase 1. Furthermore, when there exists a path P in $\mathcal{P}$ at the beginning of Phase 1, then there cannot be such a subset of cycles in $\mathcal{C}$. To see this, let p denote the length of the path P. Then, the LCMP-approximation algorithm considers cycles in $\mathcal{C}$ only if $z > p$. This implies that the matching M' has size $|M'| = |M| + 1 \geq |M_{\text{opt}}| + 1$. If there would exist cycles $C_1, \dots, C_l$ in $\mathcal{C}$ with total length z', then the LCMP-approximation algorithm would return a matching M_{app} with $|M_{\text{app}}| = |M'| \geq |M_{\text{opt}}| + 1$, which contradicts the optimality of M_{opt}.

7.2. The objective branching method

In Section 7.1 we presented an approximation algorithm for the LCMP on level graphs. The algorithm we develop in this section is designed to be applicable to perfect matching problems with an additional equality constraint which can refer to any subset of the edges. It *exactly* solves the equality constrained perfect matching problem (ECPMP) (see Section 4.1). We call our algorithm the *objective branching algorithm.*

We recall that the problem which is solved by the objective branching algorithm is defined as follows. Let $G = (U \cup V, E)$ be a bipartite graph with $|U| = |V| = n$ and $E = \{e_1, \ldots, e_m\}$. Let R be any subset of E and let k be a nonnegative integer with $k \leq n$. The ECPMP is the problem of finding a feasible solution to the system

$$\sum_{e \in \delta(u)} x_e = 1 \qquad \forall\, u \in U \tag{7.8}$$

$$\sum_{e \in \delta(v)} x_e = 1 \qquad \forall\, v \in V \tag{7.9}$$

$$\sum_{e \in R} x_e = k \tag{7.10}$$

$$x_e \in \{0,1\} \qquad \forall\, e \in E. \tag{7.11}$$

7.2.1. Motivation of the objective branching algorithm

Our algorithm makes use of the structure of ECPMP instances. The central idea is to take advantage of the fact that except for the side constraint (7.10), the problem consists of the well-known classical perfect matching problem ((7.8),(7.9),(7.11)). It is crucial for the objective branching algorithm that there exists a polynomial-time solution algorithm for the weighted version of this subproblem, i.e. for the maximum weight perfect matching problem (w-PM):

$$\begin{aligned} \max \quad & \sum_{i=1}^{m} c_i x_i \\ \text{s.t.} \quad & (7.8), (7.9), (7.11). \end{aligned}$$

We refer to the Hungarian algorithm (see Algorithm B.2 in Appendix B) as a polynomial solution method for the weighted perfect matching problem on bipartite graphs. Note that the input graph G must be extended to a complete bipartite graph by adding 0-weight edges. Further, the edge weights c_e must be incremented by a constant in order to ensure that the perfect matching contains edges in E only.

The way we make use of w-PM when solving the ECPMP is motivated by the following idea. Each solution $x^* \in \{0,1\}^m$ of the ECPMP is a feasible point for w-PM. Hence, for determining a solution of the ECPMP it suffices to consider the points $x \in \{0,1\}^m$ which are feasible for w-PM. Further, each feasible point $x \in \{0,1\}^m$ for w-PM is a vertex of the m-dimensional unit cube $[0,1]^m$. For each such x there exists a coefficient vector $c \in \{-1,1\}^m$ such that x is the unique optimal solution to w-PM with objective function coefficient vector c. Thus, also for each solution $x^* \in \{0,1\}^m$ of the ECPMP there exists a coefficient vector $c \in \{-1,1\}^m$ such that x^* is the unique optimal solution to w-PM with objective function coefficient vector c.

On the other hand, if a feasible point x of w-PM satisfies the constraint (7.10), then x solves the ECPMP. For that reason, we call a feasible point of w-PM a *candidate solution* for the ECPMP.

7.2.2. Main strategy of the objective branching algorithm

In order to get candidate solutions for the ECPMP, we solve instances of w-PM with different objective function coefficient vectors $c \in \{-1, 0, 1\}^m$. An optimal solution x of such a w-PM instance, which is returned by the weighted perfect matching solver, is considered as a candidate solution for the ECPMP, and it is checked whether $\sum_{e \in R} x_e = k$. If this equality holds, the additional side constraint (7.10) is satisfied and we have found a solution to the ECPMP. If the additional side constraint is not satisfied, the objective branching algorithm continues by solving the next w-PM instance. We allow 0 as a value of an objective function coefficient in order to be able to formulate w-PM instances where the set of optimal solutions we aim for is not only a single vertex of the unit cube but rather the set of vertices of a face of the unit cube.

The problem instances of w-PM are organized in a binary tree structure T. Each node in T corresponds to a w-PM problem instance with a specific objective function. Let PM(c) denote the w-PM instance with given objective function coefficient vector $c \in \{-1, 0, 1\}^m$. In the following, we also use the term PM(c) to name the node which stands for the problem instance PM(c). We use common notations concerning binary trees: The *depth* of a node in T is the length of the path from the root of T to that node, and a *level* of T consists of all nodes with the same depth. We denote the depth of the instance PM(c) by $l(c)$ for all instances PM(c) in T. So, an instance PM(c) is at level i if and only if $l(c) = i$. For a vector $c \in \{-1, 0, 1\}^m$ we define

$$
\begin{aligned}
N_c &:= \{i = 1, \dots, m \mid c_i = -1\}, \\
Z_c &:= \{i = 1, \dots, m \mid c_i = 0\}, \\
P_c &:= \{i = 1, \dots, m \mid c_i = 1\}.
\end{aligned}
$$

The root node of T stands for the problem PM($\mathbf{0}$), i.e. the w-PM instance where the coefficient vector of the objective function is $(0, \dots, 0)^\top \in \mathbb{R}^m$. If the candidate solution coming from a problem PM(c) does not solve the ECPMP instance, then two child nodes are created. A node PM(c) with $0 \le l(c) \le m - 1$ can have two child nodes PM(c^-) and PM(c^+), with

$$
c_i^- := \begin{cases} c_i & \text{if } i \neq l(c) + 1, \\ -1 & \text{if } i = l(c) + 1, \end{cases} \qquad c_i^+ := \begin{cases} c_i & \text{if } i \neq l(c) + 1, \\ 1 & \text{if } i = l(c) + 1. \end{cases}
$$

We call the resulting tree T an *objective branching tree.* Figure 7.2 illustrates the general construction scheme of T. One can observe that for any problem instance PM(c) at level l it holds that $c_i \in \{-1, 1\}$ for all $i = 1, \dots, l$ and $c_i = 0$ for all $i = l + 1, \dots, m$. With each node PM(c) in T we associate a set

$$
F(c) = \{x \in \{0, 1\}^m \mid x \text{ satisfies (7.8) and (7.9)}, x_i = 0 \ \forall \ i \in N_c, x_j = 1 \ \forall \ j \in P_c\}.
$$

It holds that $F(c^-), F(c^+) \subseteq F(c)$ for all coefficient vectors c of instances PM(c) with depth $l(c) \le m - 1$. Hence, applying this property recursively yields

$$
F(c') \subseteq F(c) \text{ for all nodes PM}(c') \text{ in the subtree rooted at node PM}(c). \tag{7.12}
$$

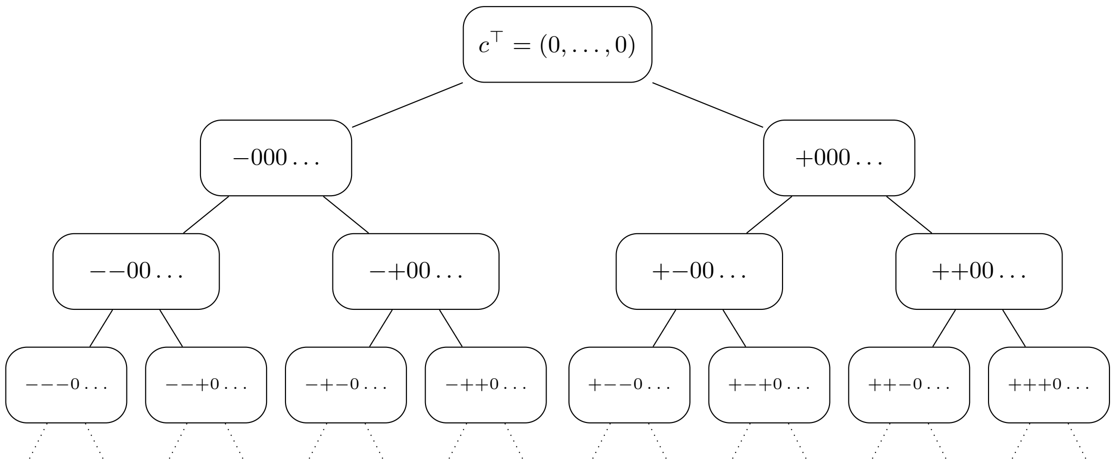


Figure 7.2.: General structure of an objective branching tree T. Each node contains the coefficient vector c of the problem instance PM(c) it stands for. Coefficient vectors are denoted in form $c_1c_2c_3\dots$ with the "$-$" sign representing a -1 and the "$+$" sign representing a 1.

Solving an instance PM(c) with a coefficient vector $c \in \{-1,0,1\}^m$ does not only yield a solution which serves as a candidate solution for the ECPMP, but can also reveal a set of vertices of the m-dimensional unit cube which can be excluded from being solutions of the ECPMP. Thus, there is the possibility to prune the objective branching tree T at some nodes.

As an essential component of our algorithm, at each node in T it is checked whether the tree T can be pruned at this node after the corresponding w-PM instance is solved. If it can be pruned, no branching takes place at the corresponding node, i.e. the current node does not have any child nodes. This way, the number of nodes and problem instances of w-PM in T which are to be solved is decreased, which is beneficial for the running-time of the objective branching algorithm. Reasons for pruning T at a node are presented in Section 7.2.3.

Taken all together, the objective branching algorithm combines solving w-PM instances to find candidate solutions for the ECPMP and constructing the objective branching tree by branching and pruning steps. It is listed as Algorithm 7.2.

7.2.3. Pruning the objective tree

In the following, we describe three conditions, each of which allows us to prune the tree T at the current node. The pruning steps are illustrated at an example in Figure 7.3. For all conditions described next, let PM($\bar{c}$) be the problem instance which was solved last and let $\bar{x}$ be an optimal solution of it.

Algorithm 7.2 Algorithm for solving the equality constrained perfect matching problem

```
ObjectiveBranching$(G, R, k)$
Input: Bipartite graph $G = (U \cup V, E)$ with $|U| = |V| = n$ and $m := |E|$, a set $R \subseteq E$,
    an integer $k$ with $0 \leq k \leq n$
Output: Perfect matching $M$ in $G$ with $|M \cap R| = k$
 1: $T \leftarrow ((0, \dots, 0)^\top)$                          ▷ Initialize list $T$ with $\mathbf{0} \in \mathbb{R}^m$
 2: $l \leftarrow 0$                                             ▷ Current level in objective branching tree
 3: $d \leftarrow \dim(P_{\text{perfect matching}}(G))$
 4: $F \leftarrow \emptyset$                                     ▷ Set of aff. indep. vectors (for Pruning Cond. 3)
 5: repeat
 6:     $c \leftarrow$ first vector in $T$; remove first vector from $T$
 7:     if $c_{l+1} \neq 0$ then
 8:         $l \leftarrow l + 1$
 9:     $x \leftarrow$ Hungarian$(G, c)$                         ▷ Where $G$ is extended to compl. bip. graph, $c$ extended
10:     if $\sum_{e \in R} x_e = k$ then                         ▷ Check feasibility of candidate solution
11:         return $x$
12:     if prune$(c, x, k, l, R, d, F) = \mathit{false}$ then    ▷ Check pruning conditions
13:         if $l \leq m - 1$ then
14:             $c^- \leftarrow c$; $c^-_{l+1} \leftarrow -1$    ▷ Coeff. vector of first child node
15:             $c^+ \leftarrow c$; $c^+_{l+1} \leftarrow 1$     ▷ Coeff. vector of second child node
16:             add $c^-$ and $c^+$ to the end of $T$
17: until $T$ is empty
18: return "problem infeasible"

19: procedure prune$(c, x, k, l, R, d, F)$
20:     $N_c \leftarrow \{i \mid c_i = -1\}$; $Z_c \leftarrow \{i \mid c_i = 0\}$; $P_c \leftarrow \{i \mid c_i = 1\}$
21:     $I_R \leftarrow \{i \mid e_i \in R\}$
22:     if $\exists\, i \in N_c : x_i = 1$ or $\exists\, j \in P_c : x_j = 0$ then    ▷ Check Pruning Cond. 1
23:         return $\mathit{true}$
24:     else if $|P_c \cap I_R| > k$ or $|P_c \cap I_R| < k - |Z_c \cap I_R|$ then    ▷ Check Pruning Cond. 2
25:         return $\mathit{true}$
26:     else                                                     ▷ Check Pruning Cond. 3
27:         if $|F| = d + 1$ then
28:             $A \leftarrow \begin{pmatrix} v^1 & \cdots & v^{d+1} \\ 1 & \cdots & 1 \end{pmatrix}$, with $\{v^1, \dots, v^{d+1}\} = F$
29:             $\pi \leftarrow$ permutation corresp. to row permutations in LU-decomposition of $A$
30:             if $\pi(i) \in N_c \cup P_c \cup \{m + 1\} \forall\, i = 1, \dots, d + 1$ then
31:                 return $\mathit{true}$
32:         else if $F \cup \{x\}$ is a set of affinely independent vectors then
33:             $F \leftarrow F \cup \{x\}$                      ▷ Update set of aff. indep. vectors
34:     return $\mathit{false}$
```

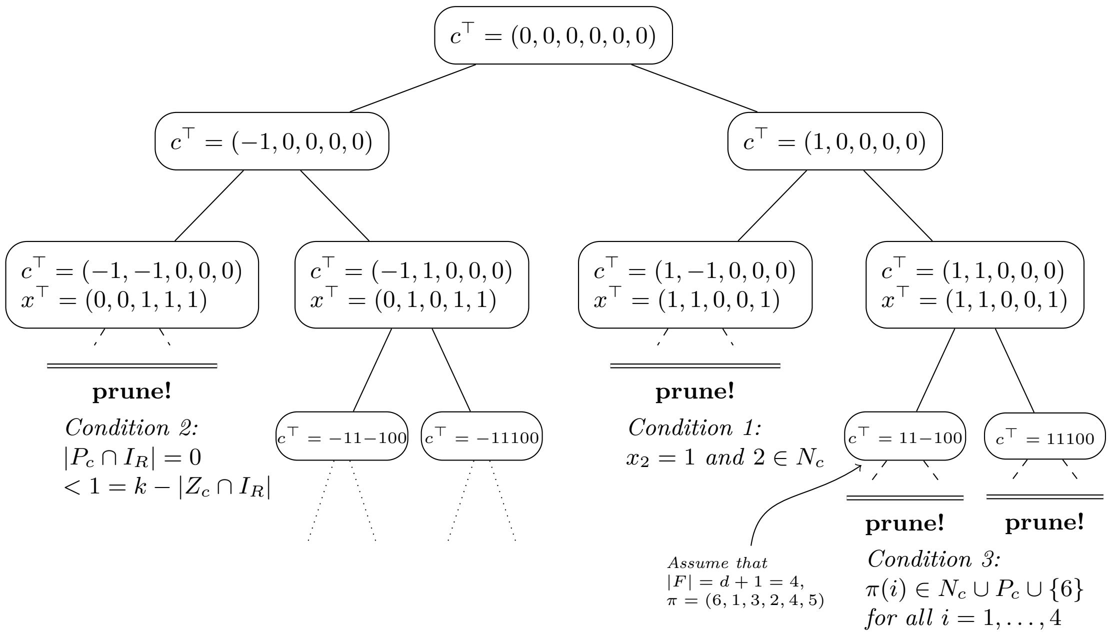

$c^\top = (0,0,0,0,0,0)$
$c^\top = (-1,0,0,0,0)$
$c^\top = (1,0,0,0,0)$
$c^\top = (-1,-1,0,0,0)$ $x^\top = (0,0,1,1,1)$
$c^\top = (-1,1,0,0,0)$ $x^\top = (0,1,0,1,1)$
$c^\top = (1,-1,0,0,0)$ $x^\top = (1,1,0,0,1)$
$c^\top = (1,1,0,0,0)$ $x^\top = (1,1,0,0,1)$
prune!
Condition 2: $|P_c \cap I_R| = 0 < 1 = k - |Z_c \cap I_R|$
$c^\top = -11-100$
$c^\top = -11100$
prune!
Condition 1: $x_2 = 1$ and $2 \in N_c$
$c^\top = 11-100$
$c^\top = 11100$
prune!
prune!
Assume that $|F| = d+1 = 4$, $\pi = (6,1,3,2,4,5)$
Condition 3: $\pi(i) \in N_c \cup P_c \cup \{6\}$ for all $i = 1,\dots,4$

Figure 7.3.: Objective branching tree T with exemplary pruning steps. The corresponding ECPMP instance is $(G = (U \cup V, E), R, k)$ with $E = \{e_1, \dots, e_5\}$, $R = \{e_1, e_2, e_5\}$, $k = 2$ and $d = \dim(P_{\text{perfect matching}}) = 3$.

Pruning Condition 1

Theorem 7.5 (Pruning Condition 1). *If $\bar{x}_i = 1$ for some $i \in N_{\bar{c}}$ or $\bar{x}_j = 0$ for some $j \in P_{\bar{c}}$, then T can be pruned at the current node.*

Proof. The set of feasible solutions of the ECPMP is a subset of the set of feasible points of w-PM. As $\bar{x}$ is feasible for w-PM and maximizes the objective function $\bar{c}^\top x$, there are no feasible solutions $\bar{x}'$ for w-PM – and thus, neither for the ECPMP – with $\bar{c}^\top \bar{x}' > \bar{c}^\top \bar{x}$. This implies that there is no feasible solution $\bar{x}'$ for w-PM with $\bar{x}'_i = 0$ for all $i \in N_{\bar{c}}$ and $\bar{x}'_j = 1$ for all $j \in P_{\bar{c}}$. Thus, $F(\bar{c}) = \emptyset$. Property (7.12) then yields that for each node PM(c') in the subtree rooted at the current node PM$(\bar{c})$ it holds that $F(c') = \emptyset$.

For each feasible point x of w-PM there is a coefficient vector $c \in \{-1,0,1\}^m$ and a problem instance PM(c) in T such that $x \in F(c)$. Hence, if $F(c) = \emptyset$ for a coefficient vector $c \in \{-1,0,1\}^m$, then the problem instance PM(c) does not need to be considered for calculating a candidate solution, i.e. the problem w-PM with weights c_i does not need to be solved. So, all nodes PM(c') in the subtree rooted at the current node PM$(\bar{c})$ can be neglected and the tree T can be pruned at the current node. □

Pruning Condition 2

Theorem 7.6 (Pruning Condition 2). *Let $I_R := \{i = 1, \dots, m \mid e_i \in R\}$ be the support of the constraint (7.10).*

If $|P_{\bar{c}} \cap I_R| > k$ *or* $|P_{\bar{c}} \cap I_R| < k - |Z_{\bar{c}} \cap I_R|$*, then* T *can be pruned at the current node.*

Proof. Due to property (7.12), it holds that if $F(\bar{c})$ contains no point which satisfies the side constraint (7.10), then none of the sets $F(c')$ for any node $\mathrm{PM}(c')$ in the subtree rooted at $\mathrm{PM}(\bar{c})$ contains a point which satisfies (7.10). We show that the negation of the pruning condition is necessary for $F(\bar{c})$ to contain points which satisfy (7.10).

Let $x \in F(\bar{c})$ be a point satisfying (7.10). It holds that

$$\begin{aligned} & P_{\bar{c}} \subseteq \{i = 1, \dots, m \mid x_i = 1\} \subseteq P_{\bar{c}} \cup Z_{\bar{c}} \\ \Rightarrow \quad & P_{\bar{c}} \cap I_R \subseteq \{i = 1, \dots, m \mid x_i = 1\} \cap I_R \subseteq (P_{\bar{c}} \cup Z_{\bar{c}}) \cap I_R. \end{aligned}$$

This implies that

$$|P_{\bar{c}} \cap I_R| \leq |\{i = 1, \dots, m \mid x_i = 1\} \cap I_R| \leq |P_{\bar{c}} \cap I_R| + |Z_{\bar{c}} \cap I_R|.$$

Due to the side constraint (7.10) we have that

$$|\{i = 1, \dots, m \mid x_i = 1\} \cap I_R| = \sum_{i \in I_R} x_i = \sum_{e \in R} x_e = k.$$

Thus, if $F(\bar{c})$ contains a point satisfying (7.10), then

$$|P_{\bar{c}} \cap I_R| \leq k \leq |P_{\bar{c}} \cap I_R| + |Z_{\bar{c}} \cap I_R|.$$

□

Refinement of Pruning Condition 2

In order to state a refinement of Pruning Condition 2 we make the following assumptions. Let $I_R := \{i = 1, \dots, m \mid e_i \in R\}$ be the support of the constraint (7.10). Let the current problem instance $\mathrm{PM}(\bar{c})$ be replaced by the two problem instances $\mathrm{PM}(c^1)$ and $\mathrm{PM}(c^2)$, with

$$c_i^1 = \begin{cases} -1 & \text{if } i \in N_{\bar{c}}, \\ 1 & \text{if } i \in P_{\bar{c}}, \\ 0 & \text{if } i \in Z_{\bar{c}} \cap I_R, \\ \delta & \text{if } i \in Z_{\bar{c}} \setminus I_R, \end{cases} \qquad c_i^2 = \begin{cases} -1 & \text{if } i \in N_{\bar{c}}, \\ 1 & \text{if } i \in P_{\bar{c}}, \\ 0 & \text{if } i \in Z_{\bar{c}} \cap I_R, \\ -\delta & \text{if } i \in Z_{\bar{c}} \setminus I_R, \end{cases}$$

where $\delta > 0$ is sufficiently small such that $x'_i = \bar{x}_i$ for all optimal solutions x' of $\mathrm{PM}(c^1)$ or $\mathrm{PM}(c^2)$ and for all $i \in N_{\bar{c}} \cup P_{\bar{c}}$.

Let x^1 and x^2 be optimal solutions of $\mathrm{PM}(c^1)$ and $\mathrm{PM}(c^2)$, respectively. Further, we assume that $x^1 \in F(c^1)$ and $x^2 \in F(c^2)$, as otherwise the objective branching tree can be pruned at the current node due to Pruning Condition 1.

Theorem 7.7 (Pruning Condition 2^{refd})**.** *If* $\sum_{i \in I_R} x_i^1 > k$ *or* $\sum_{i \in I_R} x_i^2 < k$*, then* T *can be pruned at the current node.*

Proof. The coefficient vector c^1 corresponds to the coefficient vector $\bar{c}$ where all components with value 0 and with indices not in the support of the additional constraint have the value δ. The point $x^1 \in F(c^1)$ is an optimal solution of $\mathrm{PM}(c^1)$, which is a maximum weight problem. Due to the choice of the values in the coefficient vector c^1, the point x^1 contains the maximum number of components x_e with value 1 for $e \in Z_{\bar{c}} \setminus I_R$, among all points $x \in F(c^1)$. Thus, it holds that $\sum_{i \in I_R} x_i^1 \leq \sum_{i \in I_R} x_i$ for any point $x \in F(c^1)$. As a consequence, $\sum_{i \in I_R} x_i^1$ is the minimum value of the left-hand side of the constraint (7.10), evaluated at any point $x \in F(c^1)$. The case for the problem instance $\mathrm{PM}(c^2)$ and the inequality $\sum_{i \in I_R} x_i^2 < k$ is treated analogously. □

Pruning Condition 3

In order to state Pruning Condition 3 we make the following assumptions. Let d be the dimension of the perfect matching polytope of w-PM (see Theorem 1.20). Let $v^1, \ldots, v^{d+1} \in \{0,1\}^m$ be feasible points for w-PM which are affinely independent. We define the matrix A as

$$A := \begin{pmatrix} v^1 & \cdots & v^{d+1} \\ 1 & \cdots & 1 \end{pmatrix} \in \{0,1\}^{(m+1)\times(d+1)}.$$

Let $LU = PA$ be an LU-decomposition of the matrix A, where $L \in \mathbb{R}^{(m+1)\times(d+1)}$ and $U \in \mathbb{R}^{(d+1)\times(d+1)}$. The matrix $P \in \{0,1\}^{(m+1)\times(m+1)}$ is a permutation matrix. Let $\pi : \{1, \ldots, m+1\} \to \{1, \ldots, m+1\}$ be the permutation corresponding to the row permutations resulting from multiplication of P from the left.

Theorem 7.8 (Pruning Condition 3)**.** *If $\pi(i) \in N_{\bar{c}} \cup P_{\bar{c}} \cup \{m+1\}$ for all $i = 1, \ldots, d+1$, then T can be pruned at the current node.*

Proof. Each feasible point $x \in \{0,1\}^m$ of w-PM can be expressed as an affine combination of the points $v^1, \ldots, v^{d+1}$. Hence, there exists a vector $\lambda \in \mathbb{R}^{d+1}$ such that $A\lambda = \binom{x}{1}$.

Using the LU-decomposition from above, the equality system $A\lambda = \binom{x}{1}$ can be written as

$$LU\lambda = P \begin{pmatrix} x \\ 1 \end{pmatrix}. \tag{7.13}$$

The equality system (7.13) has a solution if and only if there exists a vector $z \in \mathbb{R}^{d+1}$ such that

$$Lz = P \begin{pmatrix} x \\ 1 \end{pmatrix}. \tag{7.14}$$

As the matrix L results from an LU-decomposition, it is of the form $L = \binom{L^1}{L^2}$, with $L^1 \in \mathbb{R}^{(d+1)\times(d+1)}$ being a lower triangular matrix and $L^2 \in \mathbb{R}^{(m-d)\times(d+1)}$.

Let $y := P\left(\begin{smallmatrix}x\\1\end{smallmatrix}\right)$, i.e. $y_i = x_{\pi(i)}$ for all $i = 1, \ldots, m$ with $\pi(i) \neq m+1$ and $y_i = 1$ if $\pi(i) = m+1$. Due to the fact that L^1 has full rank, the equality system

$$L^1 z = \begin{pmatrix} y_1 \\ \vdots \\ y_{d+1} \end{pmatrix} \tag{7.15}$$

has a unique solution z^*. Hence, the equality system (7.14) (and therefore (7.13)) is solvable if and only if

$$L^2 z^* = \begin{pmatrix} y_{d+2} \\ \vdots \\ y_{m+1} \end{pmatrix}.$$

This has the following consequence. If the values of the components $x_{\pi(i)}$ are fixed for all $i = 1, \ldots, d+1$ with $\pi(i) \neq m+1$, then all values $y_1, \ldots, y_{d+1}$ in (7.15) are fixed as well, and there exists at most one feasible point for w-PM.

For the current problem instance PM($\bar{c}$) we are only interested in feasible solutions that are in $F(\bar{c})$, as all other feasible solutions are in sets $F(c')$ with $c'_i \neq \bar{c}_i$ for some $i = 1, \ldots, l(\bar{c})$. Therefore, when $\pi(i) \in N_{\bar{c}} \cup P_{\bar{c}} \cup \{m+1\}$ for all $i = 1, \ldots, d+1$, the values of the components $x_{\pi(i)}$ are fixed for all $i = 1, \ldots, d+1$. With these values fixed, there is at most one feasible point for w-PM, which implies that $|F(\bar{c})| \leq 1$. Due to the property (7.12), after solving the current instance PM($\bar{c}$), none of the nodes in the subtree rooted at node PM($\bar{c}$) need to be considered and the objective branching tree T can be pruned at the current node. □

A special property of Pruning Condition 3 is that once it is fulfilled for a node PM($\bar{c}$) with depth $l(\bar{c})$, the condition is also fulfilled for all nodes PM(c') with $l(c') \geq l(\bar{c})$.

7.2.4. Notes on the implementation of the objective branching algorithm

This section is concluded with remarks on the implementation of the developed algorithm, which leads to a lower number of nodes in the objective branching tree in practice. We also implemented the objective branching algorithm as a C++ program to retrieve experimental results on its practical behavior. A table with information on the structural behavior of the algorithm, i.e. the number of nodes in the objective branching tree and the number of pruning decisions, is given in Appendix B.3. Its data is based on runs of the algorithm on different instances of the LCPMP, where for each value of the parameter of the level constraint we considered 100 randomly created bipartite level graphs.

General notes on the implementation of pruning conditions

Concerning the implementation of the objective branching algorithm, the pruning conditions can be handled as independent components. This makes it possible to run the algorithm with only a selected subset of pruning conditions, in order to investigate the effect of special combinations of these conditions. One should note that even when all pruning conditions are deactivated, the algorithm returns a correct solution. The modular arrangement also offers the possibility to extend the set of pruning conditions with further conditions. In the following we present additional information on the implementation of pruning conditions.

Implementation of Pruning Condition 2

Pruning Condition 2 does not depend on the optimal solution $\bar{x}$ of PM($\bar{c}$). Therefore, this condition can be checked before an optimal solution to the current problem instance is calculated. Then, each pruning instruction it returns also avoids the calculation of an optimal solution to the current instance of w-PM.

Implementation of Pruning Condition 2^{refd}

It is not possible to apply the refined version of Pruning Condition 2 before the current w-PM instance is solved, as Pruning Condition 2^{refd} makes use of the optimal solution of this instance.

Pruning Condition 2^{refd} is fulfilled if $\sum_{i \in I_R} x_i^1 > k$ or $\sum_{i \in I_R} x_i^2 < k$. It is implemented to first check if $\sum_{i \in I_R} x_i^1 > k$. In order to do this, only the result x^1 of the problem instance PM(c^1) is needed. In the case that for x^1 the condition $\sum_{i \in I_R} x_i^1 > k$ is already satisfied, the second part of the pruning condition, which is to check whether $\sum_{i \in I_R} x_i^2 < k$ holds true, is not needed. Thus, the solution x^2 of the problem instance PM(c^2) is not needed to be known.

We make use of this fact and insert the problem instance PM(c^2) into the objective branching tree if and only if $\sum_{i \in I_R} x_i^1 \leq k$. This reduces the total number of nodes inserted into the tree, which is beneficial for the practical running-time of the objective branching algorithm.

Implementation of Pruning Condition 3

For applying Pruning Condition 3, it is required that $d+1$ affinely independent points $v^1, \dots, v^{d+1}$, all feasible for w-PM, are known. In Algorithm 7.2, these nodes are collected in the set F. As long as F contains less than $d+1$ elements, for each solution x returned by the w-PM solver it is checked whether the points in F together with the point x are affinely independent. If they are, F is expanded by x. In order to determine whether the points in $F = \{v^1, \dots, v^s\}$ and x are affinely independent, the linear equation system $\begin{pmatrix} v^1 & \cdots & v^s \\ 1 & \cdots & 1 \end{pmatrix} \lambda = \mathbf{0}$ is solved. Using an LU-decomposition for solving this linear equation system is advantageous in two ways.

First, when a new point v^{s+1} is added to F, the new LU-decomposition does not need to be calculated from the beginning, but rather can be updated; i.e. it can be calculated based on the former LU-decomposition. When F contains $d+1$ elements, then the LU-decomposition used for deciding affine independence also is the LU-decomposition which is used in Pruning Condition 3.

Second, the values $\pi(1),\dots,\pi(s)$ of the row permutation in the LU-decomposition of the matrix $A = \begin{pmatrix} v^1 & \cdots & v^s \\ 1 & \cdots & 1 \end{pmatrix}$ with $s \leq d$ do not change when the LU-decomposition is updated due to a new point v^{s+1}. We make use of this fact by bringing forward components of the coefficient vector based on which two new child nodes are created: We first branch on those indices j for which there exists an $i = 1,\dots,s$ such that $\pi(i) = j$. This way, Pruning Condition 3 is fulfilled earlier in the course of the algorithm.

Let $\bar{I} := \{i = 1,\dots,m \mid c_i = 0 \text{ for all } \mathrm{PM}(c) \in T\}$ be the set of indices i for which no node $\mathrm{PM}(c)$ with $c_i = \pm 1$ currently exists in T. This implies that for each index $i \in \bar{I}$ no instance $\mathrm{PM}(c)$ with $c_i = \pm 1$ has been solved yet. All indices in $\bar{I}$ can be brought forward as branching indices when a new level in the objective tree is filled with nodes, as they have not been used as index for branching steps yet.

8. Summary

The two main problems which were subject of our investigations in this work were the *couple constrained matching problem* and the *level constrained matching problem*. They are representatives of the class of resource constrained matching problems – a problem class which consists of matching problems with additional side constraints.

For both problems we presented their formulations and introduced the corresponding assignment problem with the same type of additional side constraints. Furthermore, connections between the problems discussed in this work and related combinatorial optimization problems from the literature were shown.

The types of additional side constraints in the couple constrained matching problem and the level constrained matching problem differ in the number of imposed constraints as well as in their structure. We related the two problems to each other and presented a polynomial reduction from the level constrained perfect matching problem to the couple constrained perfect matching problem. This implies that the level constrained matching problem can be regarded as a special case of the couple constrained matching problem.

As a central result we showed that the couple constrained matching problem is NP-hard. To prove this result, we first generalized a complexity result from the literature, which is that the problem of weighted matching with bonds is NP-hard. Then, we provided some complexity results for intermediate problems, which finally enabled us to show the complexity of the couple constrained matching problem. Furthermore, we strengthened our result by proving that the problem stays NP-hard even when requiring the underlying graph to be a bipartite cycle. This could be shown as all reduction steps from the problem matching with bonds to the couple constrained matching problem were designed such that the constructed problem instances in the reductions were stated on bipartite graphs only.

A substantial part of this work dealt with complexity aspects of the level constrained matching problem and the polynomially equivalent level constrained perfect matching problem. We proved that the level constrained perfect matching problem is polynomially equivalent to three other combinatorial optimization problems from the literature, whose computational complexity is unknown. One of these problems was the resource constrained perfect matching problem, for which we investigated the impact of fixed input parameters on its complexity. To this end, we presented different settings of fixed and variable input parameters which were sufficient for the problem to become polynomially solvable or NP-hard.

Along our complexity analysis of the couple constrained matching problem and the level constrained matching problem we also considered assignment problems with either a set of additional couple constraints or an additional level constraint imposed on it.

A problem of major relevance we introduced was the couple and level constrained matching problem with on-level couples. In this problem both types of additional side constraints were brought together. We proved that the decision version of this problem is NP-complete by polynomially reducing the decision version of the clique problem to it. In this problem reduction we made use of a modular approach for constructing and analyzing the underlying graph of the couple and level constrained matching problem with on-level couples. This complexity result is particularly interesting, as the problem is trivial to solve when it is formulated without the level constraint. This way we showed the impact the level constraint can have on the complexity of an easy-to-solve variant of a resource constrained matching problem.

One chapter of this work was dedicated to polytopes of resource constrained matching problems. We established new results on the structure of the level constrained perfect matching polytope in terms of a characterization of the non-integral vertices of the polytope corresponding to the linear relaxation of the level constrained perfect matching problem. This was done by presenting a set of linear inequalities which separate all non-integral vertices of this polytope from the convex hull of all integer solutions of the level constrained perfect matching problem. Furthermore, we showed that for each such non-integral vertex a separating inequality can be found in polynomial time.

As solution approaches for the level constrained matching problem, we presented two algorithms. The first algorithm we developed was a polynomial approximation algorithm for the level constrained matching problem stated on level graphs. Its approximation quality ensured that the size of a returned matching was at most one edge less than the size of an optimal solution. The second algorithm we developed was the objective branching algorithm. It solves the equality constrained perfect matching problem, which is a generalization of the level constrained perfect matching problem, to optimality. We designed the algorithm to benefit from the fact that the problem it solves contains the well-known perfect matching problem as a subproblem. A run of this algorithm is based on a tree structure in which each node is used to calculate a candidate solution of the equality constrained perfect matching problem.

The results presented in this work contribute essential parts to the understanding of the matching problem either with additional couple constraints, with an additional level constraint, or with both types of constraints imposed together. For the couple constrained matching problem we showed that there is no polynomial solution algorithm, assuming that $P \neq NP$. Regarding the level constrained matching problem we analyzed its complexity by means of complexity comparisons to other combinatorial optimization problems and to problems with different combinations of fixed input parameters. We showed that the level constraint is a crucial side constraint for the couple and level constrained matching problem with on-level couples to be NP-hard. The existence of a polynomial approximation algorithm and the characterization of non-integral vertices of its polytope might serve as a promising starting point for further research approaches towards this problem.

A. NP-completeness

In this section, we review the notion of NP-completeness. We informally present the problem classes P and NP, define the classes of NP-hard and NP-complete problems, and introduce the terminology used for developing complexity results in this work. The introduction to this topic follows the introduction to the theory of NP-completeness in [38] and [4]. For a formal definition of the concept of NP-completeness and a comprehensive discussion of computational complexity, see [26].

A.1. Running-times

When analyzing running-times, we are mostly interested in their asymptotical behavior. We use the standard notation for denoting asymptotical upper bounds, which is defined as follows: For a function $g : \mathbb{N} \to \mathbb{R}$ we define

$$\mathcal{O}(g) := \{f : \mathbb{N} \to \mathbb{R} \mid \exists\ c \in \mathbb{R}, n_0 \in \mathbb{N} \text{ such that } \forall\ n \geq n_0 : f(n) \leq c \cdot g(n)\}.$$

Let P be any problem and let I be an instance of P. The *size* $|I|$ of instance I is the binary encoding length of I. In general, the encoding scheme does not need to be binary; any scheme with at least two symbols is suitable.

We say that an algorithm *solves* the problem P, if it accepts all instances of P as input and returns a solution for each instance in finite time. When applying an algorithm to solve a problem instance I, we refer to its running-time $T(I)$ as the number of time steps (or basic computational steps) needed to produce an output. A *polynomial-time algorithm* (or just *polynomial algorithm*) for a problem P is an algorithm for which a constant k exists such that $T(I) \in \mathcal{O}(n^k)$ for all instances I of P with size n. We say that a problem P is *polynomially solvable*, if there exists a polynomial-time algorithm for P.

A.2. The classes P and NP

For the purpose of defining the sets P, NP and the classes of NP-hard and NP-complete problems, we consider all problems as *decision problems*, i.e. problems in which for any instance the answer is either "yes" or "no". As an example, we consider an instance of the matching problem on a graph G together with a nonnegative, integer-valued parameter k. An instance of the corresponding decision problem would be the question whether G contains a matching of size at least k.

Formally, given an encoding alphabet Σ with Σ^* denoting the set of all finite strings of elements in Σ, a decision problem Π is a subset $\Pi \subseteq \Sigma^*$ and an instance of Π is a string x $\in \Sigma^*$. According to this notation, the set Π consists of all instances $x \in \Sigma^*$ which have the answer "yes". Hence, instances to which the answer is "yes" can be interpreted as feasible instances of a problem. A feasible instance of a decision problem is called a *yes-instance*. An instance which is infeasible is called a *no-instance*.

The feasibility of an instance I of a decision problem P is strongly linked with a *certificate* (or *witness*) $C(I)$, which is a structure of size polynomially bounded in $|I|$, and which affirms I to be a yes-instance. In an instance $I = (G, k)$ of the decision version of the matching problem from above, a matching M in G of size $|M| \geq k$ is a certificate for I being a yes-instance.

The class P consists of all decision problems which are polynomially solvable. The class NP consists of all decision problems P for which there exists a polynomial-time algorithm which verifies that for a given yes-instance I of P and a given structure $C(I)$, the structure $C(I)$ is a correct certificate for I.

Let P and P' be two decision problems. We say that P' is *polynomially reducible* to P (denoted by $P' \leq_p P$), if there is a mapping f from the set of instances of P' to the set of instances of P such that the following holds: For each instance I' of P', the instance $f(I')$ of P is constructed in time bounded by a polynomial in $|I'|$, with the property that solving the instance $f(I')$ solves the instance I' as well. A polynomial reduction of P' to P implies that the problem P' is at most as hard to solve as the problem P. If $P' \leq_p P$ and $P \leq_p P'$, then the problems P and P' are called *polynomially equivalent*.

A decision problem P is called *NP-hard*, if $P' \leq_p P$ for all $P' \in$ NP. An NP-hard problem P which also is in NP is called NP-complete. Hence, NP-complete problems can be considered as the hardest problems in NP.

Cook [14] proved that the satisfiability problem is NP-complete. This is the first problem which was shown to be NP-complete. The existence of an NP-complete problem P allows us to prove the NP-completeness of a problem $P' \in$ NP by means of a reduction: As it holds that $P'' \leq_p P$ for all $P'' \in$ NP, one can show that P' is NP-complete by polynomially reducing P to P'.

A.3. Optimization problems

Combinatorial optimization problems often appear in the form of an optimization problem, in which an optimal structure is searched for instead of a "yes"- or "no"-answer. To each optimization problem there is a corresponding decision problem which contains an additional parameter serving as a bound for the optimization problem's objective function. It is a lower bound if the optimization problem is a maximization problem and it is otherwise an upper bound. For example, in the matching problem the task is to determine a matching of maximum size. In the decision problem corresponding to the matching problem the question is whether the graph contains a matching of size at least k, where k is the additional lower bound parameter.

Optimization problems and their corresponding decision problems are polynomially equivalent (see [26]). Due to this fact, an optimization problem is polynomially solvable if and only if the corresponding decision problem is in P.

Furthermore, we will call an optimization problem NP-hard if the corresponding decision problem is NP-complete, as the existence of a polynomial-time algorithm for the optimization problem would imply that P = NP.

A review of NP-completeness results for decision versions of some basic combinatorial optimization problems, together with the corresponding reduction sequences beginning with the satisfiability problem, is given in [38].

B. Supplementary Information on Algorithms

In this section, additional information concerning solution algorithms for flow problems and for matching problems are listed. Furthermore, we present results of runs of the objective branching algorithm applied on randomly created problem instances of preset size.

B.1. Algorithms for flow problems

Let $D = (N, A)$ be a digraph with $n := |N|$ and $m := |A|$. Let u_{ij} be the capacity of the arc (i, j) for all $(i, j) \in A$ and let $U := \max_{(i,j)\in A} u_{ij}$. Table B.1 lists solution algorithms for the maximum flow problem together with their running-times on instances of the size as above. Extensive descriptions of the algorithms can be found in [4].

Algorithm	**Running-time**
Labeling algorithm	$\mathcal{O}(nmU)$
Capacity scaling algorithm	$\mathcal{O}(nm \log(U))$
Successive shortest path algorithm	$\mathcal{O}(n^2 m)$
Generic preflow-push algorithm	$\mathcal{O}(n^2 m)$
FIFO preflow-push algorithm	$\mathcal{O}(n^3)$
Highest-label preflow-push algorithm	$\mathcal{O}(n^2 \sqrt{m})$
Excess scaling algorithm	$\mathcal{O}(nm + n^2 \log(U))$

Table B.1.: List of algorithms for the maximum flow problem (from [4]).

We now list algorithms for the minimum cost flow problem. Let c_{ij} be the cost of the arc (i, j) for all $(i, j) \in A$ and let $C := \max_{(i,j)\in A} |c_{ij}|$. Table B.2 lists solution algorithms for the minimum cost flow problem together with their running-times on instances of the size as above. Descriptions and further analyses of the algorithms can be found in [4].

Algorithm	**Running-time**
Cycle-canceling algorithm	$\mathcal{O}(mCU)$
Successive shortest path algorithm	$\mathcal{O}(nU)$
Primal-dual algorithm	$\mathcal{O}(\min\{nU, nC\})$
Capacity scaling algorithm	$\mathcal{O}((m\log(U))(m + n\log(n)))$
Cost scaling algorithm	$\mathcal{O}(n^3\log(nC))$
Double scaling algorithm	$\mathcal{O}(nm\log(U)\log(nC))$
Minimum mean cycle canceling algorithm	$\mathcal{O}(n^2m^3\log(n))$
Repeated capacity scaling algorithm	$\mathcal{O}((m^2\log(n))(m + n\log(n)))$
Enhanced capacity scaling algorithm	$\mathcal{O}((m\log(n))(m + n\log(n)))$

Table B.2.: List of algorithms for the minimum cost flow problem (from [4]).

B.2. Algorithms for matching problems

B.2.1. The maximum matching algorithm

Before presenting Edmond‘s algorithm (listed as Algorithm B.1) for solving the matching problem, we provide basic information on the idea behind the algorithm. To this end, we briefly present the concept of alternating trees and the contraction procedure. For a more detailed analysis of the algorithm and its theoretical background, see [20], [49] and [34].

Definition B.1. *Let $G = (V, E)$ be a graph and let M be a matching in G. Let $s \in V$ be an M-exposed node. An* alternating tree *T with root s is a tree T in G which has the root node s and which satisfies the following properties:*

1. *For each node $v \in V(T)$ the (s, v)-path in T is an M-alternating path.*
2. *Each node $v \in V(T) \setminus \{s\}$ is covered by an edge in $M \cap E(T)$.*

By *even*(T) and *odd*(T) we denote all the nodes in $V(T)$ whose distances to the root node s are even or odd, respectively. A basic property of an alternating tree in G concerning a matching M used by the algorithm is the following: Each node $u \in \text{even}(T)$ which is adjacent to an M-exposed node $v \in V \setminus V(T)$ induces an (s, v)-path in G which is an augmenting path concerning M.

We now describe the concept of contracting a subset of nodes in $G = (V, E)$. Let $B \subseteq V$. By G/B we denote the graph resulting from contracting the set B, i.e. identifying all nodes in B with one new vertex b. This graph then consists of the nodes $(V \setminus B) \cup \{b\}$. For all edges in G which have exactly one end-node $v \in B$, this end-node is replaced by b in the graph G/B. All edges in G with both end-nodes in B do not occur in G/B. In the algorithm, this contraction step is used to shrink a set of nodes $B \subseteq V(T)$ which build an odd cycle in G.

Algorithm B.1 Maximum matching in general graphs

```
MAXIMUM-MATCHING($G$)
Input: Graph $G = (V, E)$
Output: Maximum matching in $G$
 1: $M \leftarrow \emptyset$; $F \leftarrow \emptyset$      ▷ $F$ contains nodes of forest for which max. matching has been found
 2: while $F \neq V$ do
 3:     $G' \leftarrow G - F$      ▷ Subgraph for which max. matching will be determined
 4:     $M' \leftarrow \emptyset$      ▷ $M'$ will be max. matching in subgraph $G'$
 5:     if $M'$ is a perfect matching in $G'$ then
 6:         expand all contracted nodes in $V_T$
 7:         $F \leftarrow F \cup V_T$; $M \leftarrow M \cup M'$
 8:         goto 2
 9:     $s \leftarrow M'$-exposed node in $G'$
10:     $T = (V_T, E_T) \leftarrow (\{s\}, \emptyset)$      ▷ Start alternating tree with root $s$
11:     if $\exists\ [u, v] \in E(G')$ with $u \in \text{even}(T)$ and $v \notin \text{odd}(T)$ then
12:         if $v$ is $M'$-exposed and $v \neq s$ then
13:             augment $M'$ along the augmenting $(s, v)$-path in $T$
14:             expand $M'$ by undoing shrinking modifications      ▷ Compare Lemma B.2
15:             $G' \leftarrow G - F$      ▷ Undo shrinking applied to $G'$
16:             goto 5
17:         if $v \in \text{even}(T)$ then
18:             SHRINK($G', M', T, [u, v]$)      ▷ Shrink odd cycle
19:             goto 11
20:         if $v$ is covered by $[v, w] \in M'$ then
21:             $V_T \leftarrow V_T \cup \{v, w\}$; $E_T \leftarrow E_T \cup \{[u, v], [v, w]\}$      ▷ Let $T$ grow
22:             goto 11
23:     expand all contracted nodes in $V_T$      ▷ Situation: $T$ cannot grow, $s$ stays exposed
24:     $F \leftarrow F \cup V_T$; $M \leftarrow M \cup M'$
25:     goto 2
26: return $M$

27: procedure SHRINK($G', M', T, [u, v]$)
28:     $C \leftarrow$ cycle of odd length induced by adding $[u, v]$ to $T$
29:     $G' \leftarrow G'/V(C)$; $T \leftarrow T/V(C)$
30:     $M' \leftarrow M' \setminus E(C)$
```

Let C be a cycle of odd length in G. Lemma B.2, which can be found e.g. in [34] and [49], shows that a matching in the contracted graph $G/V(C)$ can always be extended to a matching in the original graph G.

Lemma B.2. *Let C be a cycle of odd length in G. Let $G' := G/V(C)$ and let M' be a matching in G'.*

Then, there is a matching M in the graph G with $M \subseteq M' \cup E(C)$, such that the number of M-exposed nodes in G equals the number of M'-exposed nodes in G'.

The proof of Lemma B.2 is based on the fact that $\frac{|C|-1}{2}$ edges in the cycle C can be chosen to be in the matching M. The following result can be found in [34] and [49]:

Theorem B.3. *Let $G = (V, E)$ be a graph with $n := |V|$ and $m := |E|$. Algorithm B.1 can be implemented such that its running-time on G is $\mathcal{O}(nm\alpha(n))$, where α is the inverse Ackermann function.*

For an overview of different running-times depending on the data structures used, see [49].

B.2.2. The Hungarian algorithm

The Hungarian algorithm, listed as Algorithm B.2, determines a perfect matching of maximum weight with respect to nonnegative edge weights in a complete bipartite graph $K_{n,n}$. We present the algorithm following the notations of Jungnickel in [30], where an array *mate* assigns each node i in $K_{n,n}$ a node j in $K_{n,n}$, where i and j are end-nodes of the same matching edge. The Hungarian algorithm and Theorem B.4 are due to Kuhn [35], [36].

Theorem B.4. *Algorithm B.2 determines a perfect matching of maximum weight in a complete bipartite graph $K_{n,n}$ in $\mathcal{O}(n^3)$.*

B.3. Experimental results for the objective branching algorithm

In this section, we present results of the objective branching algorithm applied to randomly created problem instances. The results give information on the behavior of the algorithm in terms of the structure of the underlying objective branching trees. Table B.3 lists these results.

B.3.1. Problem instances

The results in Table B.3 correspond to runs of the objective branching algorithm on LCPMP instances of the form (G, k), with G being a bipartite level graph and k being the parameter of the level constraint. In each LCPMP instance, the undlerying graph $G = (U \cup\!\!\!\cdot\, V)$ is a bipartite level graph with $|U| = |V| = 200$ and $|E| = 500$. This type of (sparse) graph turned out to be suitable for investigations of the practical behavior of the objective branching algorithm: Using this type of graph, there are LCPMP instances which are feasible and there are LCPMP instances which are infeasible, depending on the parameters of the level constraint. Thus, the behavior of the algorithm can be observed for both kinds of problem instances, feasible and infeasible ones.

The graphs of the LCPMP instances are constructed as follows. We build bipartite level graphs $G_l = (U \cup\!\!\!\cdot\, V, E_l)$, for $l = 1, \ldots, 100$, where $U = \{u_1, \ldots, u_{200}\}$ and

Algorithm B.2 Hungarian algorithm. Determines a maximum weight perfect matching in a complete bipartite graph.

```
HUNGARIAN($G, c$)
Input: Complete bipartite graph $G = (V, E)$ with $V = S \cup T$, where $S = \{1, \dots, n\}$
    and $T = \{1', \dots, n'\}$. Nonnegative weight $c_{ij}$ for all edges $[i, j'] \in E$.
Output: Perfect matching of maximum weight in $G$, given by array *mate*.
 1: for $v \in V$ do mate$(v) \leftarrow 0$
 2: for $i = 1$ to $n$ do $u_i \leftarrow \max\{c_{ij} : j = 1, \dots, n\}$; $v_i \leftarrow 0$
 3: nrex $\leftarrow n$
 4: while nrex $\neq 0$ do
 5:     for $i = 1$ to $n$ do $m(i) \leftarrow$ false; $p(i) \leftarrow 0$; $\delta_i \leftarrow \infty$
 6:     aug $\leftarrow$ false; $Q \leftarrow \{i \in S : \text{mate}(i) = 0\}$
 7:     repeat
 8:         remove an arbitrary node $i$ from $Q$; $m(i) \leftarrow$ true; $j \leftarrow 1$
 9:         while aug = false and $j \leq n$ do
10:             if mate$(i) \neq j'$ then
11:                 if $u_i + v_j - c_{ij} < \delta_j$ then
12:                     $\delta_j \leftarrow u_i + v_j - c_{ij}$; $p(j) \leftarrow i$
13:                     if $\delta_j = 0$ then
14:                         if mate$(j') = 0$ then
15:                             AUGMENT(mate, $p, j'$)
16:                             aug $\leftarrow$ true; nrex $\leftarrow$ nrex $- 1$
17:                         else
18:                             $Q \leftarrow Q \cup \{\text{mate}(j')\}$
19:             $j \leftarrow j + 1$
20:         if aug = false and $Q = \emptyset$ then
21:             $J \leftarrow \{i \in S : m(i) = \text{true}\}$; $K \leftarrow \{j' \in T : \delta_j = 0\}$
22:             $\delta \leftarrow \min\{\delta_j : j' \in T \setminus K\}$
23:             for $i \in J$ do $u_i \leftarrow u_i - \delta$
24:             for $j' \in K$ do $v_j \leftarrow v_j + \delta$
25:             for $j' \in T \setminus K$ do $\delta_j \leftarrow \delta_j - \delta$
26:             $X \leftarrow \{j' \in T \setminus K : \delta_j = 0\}$
27:             if mate$(j') \neq 0$ for all $j' \in X$ then
28:                 for $j' \in X$ do $Q \leftarrow Q \cup \{\text{mate}(j')\}$
29:             else
30:                 choose $j' \in X$ with mate$(j') = 0$
31:                 AUGMENT(mate, $p, j'$)
32:                 aug $\leftarrow$ true; nrex $\leftarrow$ nrex $- 1$
33:     until aug = true
34: return mate
```

```
35: procedure AUGMENT(mate, $p$, $j'$)
36:     repeat
37:         $i \leftarrow p(j)$; mate($j'$) $\leftarrow i$; next $\leftarrow$ mate($i$); mate($i$) $\leftarrow j'$
38:         if next $\neq 0$ then
39:             $j' \leftarrow$ next
40:     until next $= 0$
```

$V = \{v_1, \dots, v_{200}\}$. The edge sets E_l are sets of exactly 500 (non-parallel) edges randomly connecting nodes in U and V for all $l = 1, \dots, 100$. It is ensured that each graph G_l contains the on-level edge $[u_i, v_i]$ for all $i = 1, \dots, 200$ and $l = 1, \dots, 100$.

B.3.2. Results

The first column of Table B.3 contains the values of the parameter k for which the problem instances (G_l, k) have been solved by the objective branching algorithm for all $l = 1, \dots, 100$. The algorithm used Pruning Conditions 1, 2 and Pruning Condition 2^{refd}. For k being an entry in the first column, the other column entries in the corresponding row read as follows:

- Column **#feasible**: Number of those LCPMP instances (G_l, k), $l = 1, \dots, 100$, which are feasible.
- Column **#nodes**: Number of nodes in the objective branching tree for which the corresponding (w-PM) instances have been solved. The value presented is the mean of the values concerning all instances (G_l, k), for $l = 1, \dots, 100$.
- Column **#leaves**: Number of nodes at level 500 of the objective branching tree for which the corresponding (w-PM) instances have been solved. The value presented is the mean of the values concerning all instances (G_l, k), for $l = 1, \dots, 100$.
- Column **PC1**: Number of pruning decisions due to a satisfied Pruning Condition 1. The value presented is the mean of the values concerning all instances (G_l, k), for $l = 1, \dots, 100$.
- Column **PC2**: Number of pruning decisions due to a satisfied Pruning Condition 2. The condition is checked and possible pruning is applied before solving the (w-PM) instance corresponding to the current node. The value presented is the mean of the values concerning all instances (G_l, k), for $l = 1, \dots, 100$.
- Column **PC2$^{\text{refd}}$**: Number of pruning decisions due to a satisfied Pruning Condition 2^{refd}. The value presented is the mean of the values concerning all instances (G_l, k), for $l = 1, \dots, 100$.
- Column **#PM($\mathbf{c^2}$)**: Number of problem instances PM(c^2) solved in order to check the second part of the Pruning Condition 2^{refd}. The value presented is the mean of the values concerning all instances (G_l, k), for $l = 1, \dots, 100$.

k	**#feasible**	**#nodes**	**#leaves**	**PC1**	**PC2**	**PC2$^{\mathbf{refd}}$**	**#PM($\mathbf{c^2}$)**
0	0	1	0	0	0	1	0
5	0	1	0	0	0	1	0
10	0	1	0	0	0	1	0
15	0	1	0	0	0	1	0
20	0	1	0	0	0	1	0
25	0	1	0	0	0	1	0
30	0	1	0	0	0	1	0
35	0	1	0	0	0	1	0
40	0	1	0	0	0	1	0
45	0	1	0	0	0	1	0
50	0	1	0	0	0	1	0
55	0	1	0	0	0	1	0
60	0	1	0	0	0	1	0
65	0	1	0	0	0	1	0
70	0	1	0	0	0	1	0
75	0	1	0	0	0	1	0
80	0	1	0	0	0	1	0
85	0	1	0	0	0	1	0
90	0	1	0	0	0	1	0
95	0	1	0	0	0	1	0
100	0	1	0	0	0	1	0
105	0	1	0	0	0	1	0
110	0	1	0	0	0	1	0
115	0	1	0	0	0	1	0
120	1	1.35	0	0.04	0	1.04	0.13
125	2	1.16	0	0	0	0.98	0.08
130	3	3.09	0	0.26	0	1.18	0.81
135	8	7.22	0	0.93	0	1.15	2.53
140	13	12.05	0	1.61	0	1.29	4.51
145	27	28.95	0	4.76	0	1.28	11.32
150	41	66.38	0	12.21	0	1.66	26.05
155	63	200.09	0	43.28	0	2.31	77
160	76	261.51	0	57.71	0	2.46	100.44
165	84	351.73	0	85.71	0	2.67	131.69
170	94	386.74	0	96.63	0	3.01	144.06
175	98	423.67	0	112.61	0	3.92	154.85
180	99	282.32	0	73.14	0	4.71	104.24
185	99	173.97	0	43.02	0	4.49	65.35
190	99	198.55	0	49.01	0	8.39	74.67
195	80	232.68	0	62.66	0	9.89	84.68
200	100	2	0	0	0	0	1

Table B.3.: Results of the objective branching algorithm solving LCPMP instances.

One can see in Table B.3 that for $k = 0, 5, 10, \ldots, 115$ none of the problem instances (G_l, k) for any $l = 1, \ldots, 100$ are feasible. In other words, there is no perfect matching containing $k = 0, 5, 10, \ldots, 115$ on-level edges in the graph G_l for any $l = 1, \ldots, 100$. In this case, the objective branching algorithm terminates after solving the (w-PM) instance corresponding to the root node of the objective branching tree. This is due to Pruning Condition 2^{refd}. As there are no PM(c^2) instances considered by this pruning condition, the first part of Pruning Condition 2^{refd} is always fulfilled. Thus, it holds that $\sum_{i \in I_R} x_i^1 > k$, for I_R being the support of the level constraint and c^1 being the coefficient vector of the root node, with $c_i^1 = \delta$ for all $i \in \{1, \ldots, 500\} \setminus I_R$ and $c_i^1 = 0$, otherwise. This shows that each perfect matching in G_l contains strictly more than 115 on-level edges for all $l = 1, \ldots, 100$.

For $k = 200$, the PM(c^2) instance corresponding to the root node PM(c^1) is considered by Pruning Condition 2^{refd} during the run of the algorithm. For its coefficient vector c^2 it holds that $c_i^2 = -\delta$ for all $i \in \{1, \ldots, 500\} \setminus I_R$ and $c_i^2 = 0$, otherwise. Thus, the solution of the corresponding instance PM(c^2) yields a matching which contains all on-level edges and thus is a candidate solution that is feasible for the instance $(G_l, 200)$ for all $l = 1, \ldots, 100$.

For $k = 120, 125, 130, \ldots, 200$, there are instances (G_l, k) which are feasible for some $l = 1, \ldots, 100$. Thus, there are instances for which the objective branching tree is not pruned at the root node and a various number of nodes in the tree need to be considered until a feasible solution is found.

It turns out that the number of satisfied Pruning Conditions 1 is bounded (roughly) linearly on the number of nodes considered in the objective branching tree. Furthermore, the number of PM(c^2) instances considered by Pruning Condition 2^{refd} also depends (roughly) linearly on the number of nodes considered in the objective branching tree.

One can observe that the number of pruning decisions due to a satisfied Pruning Condition 2^{refd} increases with the value of k (except for $k = 200$) and it is independent from the number of nodes in the tree. The following relationship serves as an explanation for this behavior: For $k \geq 100$ it holds that the smaller the difference between the maximum number of on-level edges in a perfect matching (which is 200 in our instances) and the value of k, the higher the possibility that Pruning Condition 2^{refd} is satisfied.

One can see that the number of instances which are feasible increases with the value of k (except for $k = 195$). This is due to the sparse structure of the graphs G_l. Hence, the mean number number of nodes considered in the objective branching tree also increases. This trend does not continue for $k \geq 180$. The mean number of nodes considered in the objective branching tree for $k = 185$ is significantly lower than that for $k = 175$. This indicates that Pruning Condition 2^{refd} is satisfied already for nodes at lower levels in the tree when the value of k increases.

It is noticeable that no nodes at level 500 (leaf nodes) in the objective branching tree are considered by the algorithm for any instance (G_l, k). Thus, for each instance the algorithm either determines a feasible solution or determines the infeasibility of the instance before any (w-PM) instances corresponding to nodes at the maximum level of the tree have been solved.

Concerning the Pruning Condition 2 one can see that its condition is not satisfied for any instance (G_l, k). As this condition depends on the sets N_c and P_c, it is more likely to be satisfied at high tree levels. Pruning Condition 2^{refd} is the more powerful condition which is useful even at low levels of the objective branching tree. The drawback of its use is the additional PM(c^2) nodes that need to be inserted into the tree.

C. List of problem formulations

In the following we list formulations of all problems occurring in this work. The problems are grouped according to their problem type. For those problems whose formulations are given in the course of this work, their numerations are carried over. All other problem formulations are numbered separately. The formulations given in the Section Other combinatorial (optimization) problems can be found in [15] and [30].

Flow problems

Problem Formulation 1.1 (Maximum flow). *Let $D = (N, A)$ be a directed graph with a source node $s \in N$ and a sink node $t \in N$. Further, let u_{ij} be the nonnegative arc capacity of the arc (i, j) for all $(i, j) \in A$.*

The maximum flow problem *is formulated as the following linear program:*

$$\begin{aligned} \max \quad & v \\ s.t. \quad & \sum_{j:(i,j)\in A} x_{ij} - \sum_{j:(j,i)\in A} x_{ji} = \begin{cases} v & \text{if } i = s, \\ 0 & \text{if } i \in N \setminus \{s,t\}, \\ -v & \text{if } i = t \end{cases} \\ & 0 \le x_{ij} \le u_{ij} \qquad \forall\ (i,j) \in A. \end{aligned}$$

Problem Formulation 1.3 (Minimum cost flow). *Let $D = (N, A)$ be a directed graph with a source node $s \in N$ and a sink node $t \in N$. Let u_{ij} be the nonnegative arc capacity of the arc (i, j) for all $(i, j) \in A$ and let $c_{ij} \in \mathbb{R}$ be the cost value of the arc (i, j) for all arcs $(i, j) \in A$. Let $b(i)$ be the supply/demand value of the node i for all $i \in N$.*

The minimum cost flow problem *is formulated as the following linear program:*

$$\begin{aligned} \min \quad & \sum_{(i,j)\in A} c_{ij} x_{ij} \\ s.t. \quad & \sum_{j:(i,j)\in A} x_{ij} - \sum_{j:(j,i)\in A} x_{ji} = b(i) \qquad \forall\ i \in N \\ & 0 \le x_{ij} \le u_{ij} \qquad \forall\ (i,j) \in A. \end{aligned}$$

Problem Formulation 2.20 (Maximum integer equal flow). *Let $D = (V, A)$ be a digraph with two distinct nodes $s, t \in V$ denoting the source node and the sink node in D, respectively. Further, let u_{ij} be the nonnegative, integer-valued capacity of the arc (i, j) for all $(i, j) \in A$. Let $R_1, \dots, R_k$ be pairwise disjoint subsets of A.*

The maximum integer equal flow problem *is stated as the following integer linear program:*

$$\begin{aligned}
\max\quad & v \\
\textit{s.t.}\quad & \sum_{j:(i,j)\in A} x_{ij} - \sum_{j:(j,i)\in A} x_{ji} = \begin{cases} v & \textit{if } i = s, \\ 0 & \textit{if } i \in V \setminus \{s,t\}, \\ -v & \textit{if } i = t \end{cases} \\
& 0 \le x_{ij} \le u_{ij} && \forall\ (i,j) \in A \\
& x_{i_1 j_1} = x_{i_2 j_2} && \forall\ (i_1,j_1),(i_2,j_2) \in R_h, \forall\ h = 1,\dots,k \\
& x_{ij} \in \mathbb{Z} && \forall\ (i,j) \in A.
\end{aligned}$$

Classical matching problems

Problem Formulation 1.5 (Matching)**.** *Let $G = (V,E)$ be a graph.*

The matching problem *is the problem of finding a matching of maximum cardinality in G. It is formulated as the following integer linear program:*

$$\begin{aligned}
\max\quad & \sum_{e\in E} x_e \\
\textit{s.t.}\quad & \sum_{e\in\delta(v)} x_e \le 1 && \forall\ v \in V \\
& x_e \in \{0,1\} && \forall\ e \in E,
\end{aligned}$$

where $\delta(v)$ denotes the set of edges incident to node v for all $v \in V$.

Problem Formulation 1.6 (Perfect matching)**.** *Let $G = (V,E)$ be a graph.*

The perfect matching problem *is the problem of finding a matching M in G such that all nodes in G are covered by an edge in M. It is formulated as the problem of finding a feasible solution to the system*

$$\begin{aligned}
& \sum_{e\in\delta(v)} x_e = 1 && \forall\ v \in V \\
& x_e \in \{0,1\} && \forall\ e \in E.
\end{aligned}$$

Problem Formulation 1.8 (Bipartite matching)**.** *Let $G = (U \mathbin{\dot\cup} V, E)$ be a bipartite graph.*

The bipartite matching problem *is the problem of finding a matching of maximum cardinality in G.*

Problem Formulation 1.9 (Bipartite perfect matching)**.** *Let $G = (U \mathbin{\dot\cup} V, E)$ be a bipartite graph.*

The bipartite perfect matching problem *is the problem of finding a perfect matching in G.*

Problem Formulation 1.10 (Complete matching)**.** *Let $G = (U \mathbin{\dot\cup} V, E)$ be a bipartite graph.*

The complete matching problem *is the problem of finding a matching M in the bipartite graph G such that all nodes in the smaller of the two color classes U and V are covered by M. Assuming that $|U| \le |V|$, it is formulated as the problem of finding a feasible solution to the system*

$$\begin{aligned}
\sum_{e\in\delta(u)} x_e &= 1 && \forall\ u \in U\\
\sum_{e\in\delta(v)} x_e &\le 1 && \forall\ v \in V\\
x_e &\in \{0,1\} && \forall\ e \in E.
\end{aligned}$$

Problem Formulation 1.7 (Weighted matching)**.** *Let $G = (V, E)$ be a graph and let $c_e \in \mathbb{R}$ be the weight of the edge e for all $e \in E$. The* weighted matching problem *is the problem of finding a matching M in G which is of maximum weight. It is formulated as the following integer linear program:*

$$\begin{aligned}
\max \quad & \sum_{e\in E} c_e x_e \\
s.t. \quad & \sum_{e\in\delta(v)} x_e \le 1 && \forall\ v \in V\\
& x_e \in \{0,1\} && \forall\ e \in E.
\end{aligned}$$

Problem Formulation C.1 (Symmetric matching)**.** *Let $G = (U \mathbin{\dot\cup} V, E)$ be a symmetric bipartite graph with $U = \{u_1, \ldots, u_n\}$ and $V = \{v_1, \ldots, v_n\}$. A symmetric matching M in G is a matching satisfying that $[u_i, v_j] \in M$ if and only if $[u_j, v_i] \in M$ for all $i, j = 1, \ldots, n$.*

The symmetric matching problem *is the problem of finding a symmetric matching of maximum size in G.*

Problem Formulation C.2 (Symmetric perfect matching)**.** *Let $G = (U \mathbin{\dot\cup} V, E)$ be a symmetric bipartite graph with $U = \{u_1, \ldots, u_n\}$ and $V = \{v_1, \ldots, v_n\}$. A symmetric matching M in G is a matching satisfying that $[u_i, v_j] \in M$ if and only if $[u_j, v_i] \in M$ for all $i, j = 1, \ldots, n$.*

The symmetric perfect matching problem *is the problem of finding a symmetric perfect matching in G.*

Problem Formulation 1.11 (Assignment)**.** *Let n be a nonnegative integer number. Let $c_{ij} \in \mathbb{R}$ be a cost value for all $i, j = 1, \ldots, n$. The* assignment problem *(AP) is stated* *as the following integer linear program:*

$$\begin{aligned}
\min \quad & \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} & \\
s.t. \quad & \sum_{j=1}^{n} x_{ij} = 1 & \forall\ i = 1,\dots,n \\
& \sum_{i=1}^{n} x_{ij} = 1 & \forall\ j = 1,\dots,n \\
& x_{ij} \in \{0,1\} & \forall\ i,j = 1,\dots,n.
\end{aligned}$$

Problem Formulation 1.25 (3-Dimensional matching)**.** *Let $H = (U \cup V \cup W, \mathcal{E})$ be a hypergraph with pairwise disjoint color classes U, V and W, each of them of size n, and an edge set $\mathcal{E}$ where each edge is an element of $U \times V \times W$.*

The 3-dimensional matching problem (3DM) *is the problem of determining a matching $\mathcal{M}$ in H which is of size n.*

Couple constrained matching problems

Problem Formulation 2.2 (Couple constrained matching)**.** *Let $G = (V, E)$ be a graph with $V = \{v_1, \dots, v_n\}$. Let $\mathcal{F} = \{F_1, \dots, F_k\}$ be a couple collection in G.*

The couple constrained matching problem (CCMP) *is stated as the following integer linear program:*

$$\begin{aligned}
\max \quad & \sum_{e \in E} x_e & \\
s.t. \quad & \sum_{e \in \delta(v_i)} x_e \le 1 & \forall\ i = 1,\dots,n \\
& x_e = x_f & \forall\ e,f \in F, \quad \forall\ F \in \mathcal{F} \\
& x_e \in \{0,1\} & \forall\ e \in E.
\end{aligned}$$

Problem Formulation 2.3 (Couple constrained perfect matching)**.** *Let G be a graph and let $\mathcal{F}$ be a couple collection in G.*

The couple constrained perfect matching problem (CCPMP) *is the problem of finding a perfect matching satisfying the couple constraints $\mathcal{F}$ in G.*

Problem Formulation 2.5 (Couple constrained complete matching)**.** *Let G be a bipartite graph and let $\mathcal{F}$ be a couple collection in G. The* couple constrained complete matching problem (CCCMP) *is the problem of finding a complete matching satisfying the couple constraints $\mathcal{F}$ in G.*

Problem Formulation 2.25 (Weighted couple constrained matching)**.** *Let $G = (V, E)$ be a graph and let $\mathcal{F}$ be a couple collection in G. Let $c_e \in \mathbb{R}$ be the cost of the edge e for all $e \in E$.*

The weighted couple constrained matching problem (w-CCMP) *is stated as the following integer linear program:*

$$\begin{array}{lll}
\max & \sum_{e\in E} c_e x_e & \\
s.t. & \sum_{e\in\delta(v_i)} x_e \le 1 & \forall\ i = 1,\dots,n \\
& x_e = x_f & \forall\ e,f \in F, \quad \forall\ F \in \mathcal{F} \\
& x_e \in \{0,1\} & \forall\ e \in E.
\end{array}$$

Problem Formulation 2.7 (Couple constrained assignment)**.** *Let n be a nonnegative integer number. Let $\mathcal{F} = \{F_1,\dots,F_k\}$ be a set of pairwise disjoint sets of the form $F_h = \{(i,j),(k,l)\}$ with $i,j,k,l \in \{1,\dots,n\}$. Let $c_{ij} \in \mathbb{R}$ be a cost value for all $i,j = 1,\dots,n$.*

The couple constrained assignment problem (CCAP) *is stated as the following integer linear program:*

$$\begin{array}{lll}
\min & \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij} x_{ij} & \\
s.t. & \sum_{j=1}^{n} x_{ij} = 1 & \forall\ i = 1,\dots,n \\
& \sum_{i=1}^{n} x_{ij} = 1 & \forall\ j = 1,\dots,n \\
& x_{ij} = x_{kl} & \forall\ (i,j),(k,l) \in F, \quad \forall\ F \in \mathcal{F} \\
& x_{ij} \in \{0,1\} & \forall\ i,j = 1,\dots,n.
\end{array}$$

Problem Formulation 5.2 (Couple constrained matching with on-level couples)**.** *Let $G = (U \cup V, E)$ be a level graph, with $U = \{u_1,\dots,u_n\}$ and $V = \{v_1,\dots,v_n\}$. Further, let $\mathcal{F} = \{F_1,\dots,F_p\}$ be a couple collection with $F_r = \{[u_{2r-1},v_{2r-1}],[u_{2r},v_{2r}]\}$ for all $r = 1,\dots,p$.*

The couple constrained matching problem with on-level couples (CCMP-L) *is formulated as the integer linear program*

$$\begin{array}{lll}
\max & \sum_{e\in E} x_e & \\
s.t. & \sum_{e\in\delta(u_i)} x_e \le 1 & \forall\ i = 1,\dots,n \\
& \sum_{e\in\delta(v_i)} x_e \le 1 & \forall\ i = 1,\dots,n \\
& x_{2r-1,2r-1} = x_{2r,2r} & \forall\ r = 1,\dots,p \\
& x_e \in \{0,1\} & \forall\ e \in E.
\end{array}$$

Problem Formulation 3.13 (Couple constrained assignment with on-level couples)**.** *Let n be a nonnegative integer number and let p be an integer valued parameter with $1 \le p \le n$. Let $c_{ij} \in \mathbb{R}$ be a cost value for all $i,j = 1,\dots,n$.*

The couple constrained assignment problem with on-level couples (CCAP-L) *is stated as the following integer linear program:*

$$
\begin{array}{lll}
\min & \displaystyle\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} & \\
s.t. & \displaystyle\sum_{j=1}^{n} x_{ij} = 1 & \forall\ i = 1,\dots,n \\
 & \displaystyle\sum_{i=1}^{n} x_{ij} = 1 & \forall\ j = 1,\dots,n \\
 & x_{2k-1,2k-1} = x_{2k,2k} & \forall\ k = 1,\dots,p \\
 & x_{ij} \in \{0,1\} & \forall\ i,j = 1,\dots,n.
\end{array}
$$

Level constrained matching problems

Problem Formulation 2.9 (Level constrained matching)**.** *Let $G = (U \cup V, E)$ be a bipartite graph with $U = \{u_1,\dots,u_n\}$ and $V = \{v_1,\dots,v_n\}$. Further, let k be an integer with $0 \le k \le n$.*

The level constrained matching problem (LCMP) *is formulated as the following integer linear program:*

$$
\begin{array}{lll}
\max & \displaystyle\sum_{e\in E} x_e & \\
s.t. & \displaystyle\sum_{e\in\delta(u_i)} x_e \le 1 & \forall\ i = 1,\dots,n \\
 & \displaystyle\sum_{e\in\delta(v_i)} x_e \le 1 & \forall\ i = 1,\dots,n \\
 & \displaystyle\sum_{i=1}^{n} x_{[u_i,v_i]} = k & \\
 & x_e \in \{0,1\} & \forall\ e \in E.
\end{array}
$$

Problem Formulation 2.11 (Level constrained perfect matching)**.** *Let $G = (U \cup V, E)$ be a bipartite graph with $U = \{u_1,\dots,u_n\}$ and $V = \{v_1,\dots,v_n\}$. Further, let k be an integer with $0 \le k \le n$.*

The level constrained perfect matching problem (LCPMP) *is the problem of finding a perfect matching satisfying the level constraint with parameter k in G.*

Problem Formulation 2.10 (Level constrained complete matching)**.** *Let $G = (U \cup V, E)$ be a bipartite graph with $U = \{u_1,\dots,u_{n_1}\}$ and $V = \{v_1,\dots,v_{n_2}\}$. Further, let k be an integer with $0 \le k \le \min\{n_1,n_2\}$.*

The level constrained complete matching problem (LCCMP) *is the problem of finding a complete matching satisfying the level constraint with parameter k in G.*

Problem Formulation 2.27 (Level constrained symmetric perfect matching)**.** *Let* $G = (U \mathbin{\dot\cup} V, E)$ *be a symmetric bipartite graph with* $U = \{u_1, \ldots, u_n\}$ *and* $V = \{v_1, \ldots, v_n\}$*. Further, let* k *be an integer with* $0 \le k \le n$*.*

The level constrained symmetric perfect matching problem (LCPMP-SYM) *is the problem of finding a symmetric perfect matching satisfying the level constraint with parameter* k *in* G*.*

Problem Formulation C.3 (Weighted level constrained symmetric matching)**.** *Let* $G = (U \mathbin{\dot\cup} V, E)$ *be a symmetric bipartite graph with* $U = \{u_1, \ldots, u_n\}$, $V = \{v_1, \ldots, v_n\}$*. Let* $w_e \in \mathbb{R}_0^+$ *be the weight of the edge* e *for all* $e \in E$*. Further, let* k *be an integer with* $0 \le k \le n$*. A symmetric matching* M *in* G *is a matching satisfying that* $[u_i, v_j] \in M$ *if and only if* $[u_j, v_i] \in M$ *for all* $i, j = 1, \ldots, n$*.*

The weighted level constrained symmetric matching problem (w-LCMP-SYM) *is the problem of finding a symmetric matching in* G *which is of maximum weight and satisfies the level constraint with parameter* k*.*

Problem Formulation 2.13 (Level constrained assignment)**.** *Let* n *be a nonnegative integer. Let* k *be an integer with* $0 \le k \le n$*. Let* $c_{ij} \in \mathbb{R}$ *be a cost value for all* $i, j = 1, \ldots, n$*.*

The level constrained assignment problem (LCAP) *is formulated as the following integer linear program:*

$$\begin{array}{lll}
\min & \displaystyle\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} & \\
s.t. & \displaystyle\sum_{j=1}^{n} x_{ij} = 1 & \forall\ i = 1, \ldots, n \\
 & \displaystyle\sum_{i=1}^{n} x_{ij} = 1 & \forall\ j = 1, \ldots, n \\
 & \displaystyle\sum_{i=1}^{n} x_{ii} = k & \\
 & x_{ij} \in \{0, 1\} & \forall\ i, j = 1, \ldots, n.
\end{array}$$

Matching with bonds

Problem Formulation 3.4 (Matching with bonds)**.** *Let* $G = (V, E)$ *be a graph and let* $\mathcal{B}$ *be a bond structure in* G*.*

Matching with bonds (MB) *is the problem corresponding to the following integer linear* *program:*

$$\begin{aligned}
\max \quad & \sum_{e \in E} x_e && && (3.1')\\
s.t. \quad & \sum_{e \in \delta(v_i)} x_e \le 1 && \forall\ i = 1, \ldots, n \\
& x_e = x_f && \forall\ e, f \in B, \forall\ B \in \mathcal{B} \textit{ with } |B| \ge 2 \\
& x_e \in \{0, 1\} && \forall\ e \in E.
\end{aligned}$$

Problem Formulation 3.2 (Weighted matching with bonds)**.** *Let $G = (V, E)$ be a graph and let $c_e \in \mathbb{R}$ be the weight of the edge e for all $e \in E$. Let $\mathcal{B}$ be a bond structure in G.*

Weighted matching with bonds (w-MB) *is the problem corresponding to the following integer linear program:*

$$\begin{aligned}
\max \quad & \sum_{e \in E} c_e x_e \\
s.t. \quad & \sum_{e \in \delta(v_i)} x_e \le 1 && \forall\ i = 1, \ldots, n \\
& x_e = x_f && \forall\ e, f \in B, \forall\ B \in \mathcal{B} \textit{ with } |B| \ge 2 \\
& x_e \in \{0, 1\} && \forall\ e \in E.
\end{aligned}$$

Other resource constrained matching problems and related problems

Problem Formulation 4.6 (Restricted perfect matching)**.** *$G = (U \cupdot V, E)$ be a bipartite graph with $U = \{u_1, \ldots, u_n\}, V = \{v_1, \ldots, v_n\}$. Let $R_1, \ldots, R_k \subseteq E$ and let $r_1, \ldots, r_k$ be positive, integer values. The* restricted perfect matching problem (RPMP) *is the problem of finding a perfect matching M in G, which satisfies*

$$|M \cap R_i| \le r_i \quad \textit{for all } i = 1, \ldots, k.$$

Problem Formulation 4.2 (Restricted perfect matching with fixed number of restrictions)**.** *Let $G = (U \cupdot V, E)$ be a bipartite graph with $U = \{u_1, \ldots, u_n\}, V = \{v_1, \ldots, v_n\}$. Let $R_1, \ldots, R_l \subseteq E$ with l fixed. Further, let $r_1, \ldots, r_l$ be positive, integer values.*

The restricted perfect matching problem with fixed number of restrictions *(l-RPMP) is the problem of finding a perfect matching M in G, such that $|M \cap R_i| \le r_i$ for all $i = 1, \ldots, l$.*

Problem Formulation 4.3 (Exact cycle sum)**.** *Let $D = (V, A)$ be a directed graph with $V = \{v_1, \ldots, v_n\}$. Let k be an integer-valued parameter with $0 \le k \le n$.*

The exact cycle sum problem *is the problem of finding a set of node-disjoint cycles of total length exactly k in D.*

Problem Formulation 4.10 (Fixed size assignment)**.** *Let $m, n \in \mathbb{N}$ and let k be a nonnegative integer with $k \leq \min(m, n)$. Let $c_{ij} \in \mathbb{R}$ be a cost value for all $i = 1, \ldots, m$ and $j = 1, \ldots, n$.*

The fixed size assignment problem (FSAP) *is formulated as the integer linear program*

$$\begin{aligned}
\min \quad & \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} \\
s.t. \quad & \sum_{j=1}^{n} x_{ij} \leq 1 && \forall\ i = 1, \ldots, m \\
& \sum_{i=1}^{m} x_{ij} \leq 1 && \forall\ j = 1, \ldots, n \\
& \sum_{i=1}^{m}\sum_{j=1}^{n} x_{ij} = k \\
& x_{ij} \in \{0, 1\} && \forall\ i = 1, \ldots, m; j = 1, \ldots, n.
\end{aligned}$$

Problem Formulation 4.1 (Equality constrained perfect matching)**.** *Let $G = (U \uplus V, E)$ be a bipartite graph, with $U = \{u_1, \ldots, u_n\}$ and $V = \{v_1, \ldots, v_n\}$. Let $R \subseteq E$ and let k be an integer-valued parameter, with $0 \leq k \leq n$.*

The equality constrained perfect matching problem *(ECPMP) is the problem of finding a perfect matching M in G, such that $|M \cap R| = k$.*

Problem Formulation 4.11 (Equality constrained assignment)**.** *Let $K_{n,n} = (U \uplus V, E)$ be a complete bipartite graph and let $R \subseteq E$. Furhter, let k be an integer-valued parameter with $0 \leq k \leq n$.*

The equality constrained assignment problem (ECAP) *is the problem of finding a solution to the system*

$$\begin{aligned}
& \sum_{j=1}^{n} x_{ij} = 1 && \forall\ i = 1, \ldots, n \\
& \sum_{i=1}^{n} x_{ij} = 1 && \forall\ j = 1, \ldots, n \\
& \sum_{[u_i, v_j] \in R} x_{ij} = k \\
& x_{ij} \in \{0, 1\} && \forall\ i, j = 1, \ldots, n.
\end{aligned}$$

Couple and level constrained matching problem

Problem Formulation 5.4 (Couple and level constrained matching with on-level couples)**.** *Let $G = (U \uplus V, E)$ be a level graph with $U = \{u_1, \ldots, u_n\}, V = \{v_1, \ldots, v_n\}$. Let $\mathcal{F} = \{F_1, \ldots, F_p\}$ be a couple collection with $F_r = \{[u_{2r-1}, v_{2r-1}], [u_{2r}, v_{2r}]\}$ for all $r = 1, \ldots, p$. Further, let k be a nonnegative integer with $k \leq n$.*

The couple and level constrained matching problem with on-level couples (CLCMP-L) *is formulated as the integer linear program*

$$\begin{aligned}
\max \quad & \sum_{e\in E} x_e & \\
s.t. \quad & \sum_{e\in\delta(u_i)} x_e \le 1 & \forall\ i = 1,\dots,n \\
& \sum_{e\in\delta(v_i)} x_e \le 1 & \forall\ i = 1,\dots,n \\
& x_{2r-1,2r-1} = x_{2r,2r} & \forall\ r = 1,\dots,p \\
& \sum_{i=1}^{n} x_{i,i} = k & \\
& x_e \in \{0,1\} & \forall\ e \in E.
\end{aligned}$$

Other combinatorial (optimization) problems

Problem Formulation C.4 (Independent set)**.** *Let $G = (V, E)$ be a graph. A subset $S \subseteq V$ is an* independent set *in G if no two nodes in S are adjacent.*

The independent set problem (ISP) *is the problem of finding an independent set $S \subseteq V$ in G which is of maximum cardinality.*

Problem Formulation C.5 (Weighted independent set)**.** *Let $G = (V, E)$ be a graph and let $w_v \in \mathbb{R}$ be the weight of the node v for all $v \in V$. A subset $S \subseteq V$ is an* independent set *in G if no two nodes in S are adjacent.*

The weighted independent set problem (w-ISP) *is the problem of finding an independent set $S \subseteq V$ in G which is of maximum weight regarding w.*

Problem Formulation C.6 (Shortest path)**.** *Let $G = (V, E)$ be a graph and let s and t be two distinct nodes in V. Let $l_e \in \mathbb{R}_0^+$ be the length of the edge e for all $e \in E$.*

The shortest path problem (SPP) *is the problem of finding an s-t path P in G which is of minimum length.*

Problem Formulation C.7 (Chinese postman)**.** *Let $G = (V, E)$ be a graph and let $l_e \in \mathbb{R}_0^+$ be the length of the edge e for all $e \in E$. A tour $C = [v_0, e_1, v_1, \dots, e_t, v_t = v_0]$ in G with $v_i \in V$ for all $i = 0, \dots, t$ and $e_i \in E$ for all $i = 1, \dots, t$ is a* Chinese postman tour *if C contains each edge of G at least once.*

The Chinese postman problem (CPP) *is the problem of finding a Chinese postman tour C in G which is of minimum length.*

Problem Formulation C.8 (Node cover)**.** *Let $G = (V, E)$ be a graph. A subset $S \subseteq V$ is a* node cover *in G if all edges in E have at least one end-node in S.*

The node cover problem (NCP) *is the problem of finding a node cover $S \subseteq V$ in G which is of minimum cardinality.*

Problem Formulation C.9 (Clique)**.** *Let $G = (V, E)$ be a graph. A subset $S \subseteq V$ is a* clique *in G if for each pair of nodes in S the nodes are adjacent.*

The clique problem (CP) *is the problem of finding a clique $S \subseteq V$ in G which is of maximum cardinality.*

Problem Formulation C.10 (Exact cover)**.** *Let $U = \{u_i \mid i = 1, \ldots, n\}$ be a set of n elements and let $\mathcal{S} = \{S_j \mid j = 1, \ldots, m\}$ be a set of m subsets of U.*

In the exact cover problem (ECP) *the task is to find a subset of $\mathcal{S}$ which partitions the set U.*

Problem Formulation C.11 (Subset-sum)**.** *Let $U = \{u_i \mid i = 1, \ldots, n\}$ be a set of nonnegative integer numbers and let t be a nonnegative integer.*

In the subset-sum problem *the task is to find a subset S of U such that $\sum_{u \in S} u = t$.*

Problem Formulation C.12 (3-conjunctive normal form satisfiability)**.** *A* literal *in a boolean formula is an occurrence of a variable or its negation. A* clause *is a disjunction of one or more literals. A boolean formula is in* 3-conjunctive normal form*, if it is expressed as an conjunction of clauses where each clause has exactly three distinct literals.*

In the 3-conjunctive normal form satisfiability problem (3-CNF-SAT) *the task is to determine whether a given boolean function in 3-conjunctive normal form is satisfiable or not.*

Bibliography

[1] R. Aboudi, A. Hallefjord, and K. Jörnsten. A facet generation and relaxation technique applied to an assignment problem with side constraints. *European Journal of Operational Research*, 50(3):335–344, 1991.

[2] R. Aboudi and K. Jörnsten. Resource constrained assignment problems. *Discrete Applied Mathematics*, 26(2):175–191, 1990.

[3] R. Aboudi and G. L. Nemhauser. Some facets for an assignment problem with side constraints. *Operations Research*, 39(2):244–250, 1991.

[4] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms and Applications.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1993.

[5] R. K. Ahuja, J. B. Orlin, G. M. Sechi, and P. Zuddas. Algorithms for the Simple Equal Flow Problem. *Management Science*, 45(10):1440–1455, 1999.

[6] A. Y. Alfakih and K. G. Murty. Adjacency on the constrained assignment problem. *Discrete Applied Mathematics*, 87(1-3):269–274, 1998.

[7] A. Y. Alfakih, T. Yi, and K. G. Murty. Facets of an Assignment Problem with 0–1 Side Constraint. *Journal of Combinatorial Optimization*, 4(3):365–388, 2000.

[8] A. I. Ali, J. Kennington, and B. Shetty. The equal flow problem. *European Journal of Operational Research*, 36(1):107–115, 1988.

[9] G. Birkhoff. Tres observaciones sobre el algebra lineal. *Universidad Nacional de Tucuman Revista, Serie A*, 5:147–151, 1946.

[10] P. Biró and F. Klijn. Matching with couples: A multidisciplinary survey. *International Game Theory Review*, 15(2), 2013.

[11] R. E. Burkard, M. Dell'Amico, and S. Martello. *Assignment Problems.* Siam, 2009.

[12] D. G. Cattrysse and L. N. Van Wassenhove. A survey of algorithms for the generalized assignment problem. *European Journal of Operational Research*, 60(3):260–272, 1992.

[13] R. Chandrasekaran, S. N. Kabadi, and K. G. Murthy. Some NP-complete problems in linear programming. *Operations Research Letters*, 1(3):101–104, 1982.

[14] S. A. Cook. The Complexity of Theorem-proving Procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, STOC '71, pages 151–158, New York, NY, USA, 1971.

[15] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms.* The MIT Press, third edition, 2009.

[16] M. C. Costa, D. D. Werra, C. Picouleau, and B. Ries. Bicolored matchings in some classes of graphs. *Graphs and Combinatorics*, 23(1):47–60, 2007.

[17] M. Dell'Amico, A. Lodi, and S. Martello. Efficient algorithms and codes for k-cardinality assignment problems. *Discrete Applied Mathematics*, 110(1):25–40, 2001.

[18] M. Dell'Amico and S. Martello. The k-cardinality assignment problem. *Discrete Applied Mathematics*, 76(1-3):103–121, 1997.

[19] J. Edmonds. Maximum matching and a polyhedron with 0,1-vertices. *Journal of Research of the National Bureau of Standards*, 69B:125–130, 1965.

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

[21] J. Edmonds. Submodular functions, matroids, and cerain polyhedra. *Combinatorial Optimization (Edmonds Festschrift)*, pages 11–26, 2003.

[22] J. Edmonds and E. L. Johnson. Matching, Euler tours and the Chinese postman. *Mathematical Programming*, 5(1):88–124, 1973.

[23] G. Felici and M. Mecoli. Resource assignment with preference conditions. *European Journal of Operational Research*, 180(2):519–531, 2007.

[24] M. L. Fisher, R. Jaikumar, and L. N. V. Wassenhove. A multiplier adjustment method for the generalized assignment problem. *Management Science*, 32(9):1095–1103, 1986.

[25] R. Gardner, P. Gritzmann, and D. Prangenberg. On the computational complexity of reconstructing sets from their X-rays. *Discrete Mathematics*, 202(1-3):45–71, 1999.

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

[27] M. R. Garey, D. S. Johnson, and L. Stockmeyer. Some simplified np-complete graph problems. *Theoretical Computer Science*, 1(3):237–267, 1976.

[28] A. Hefner and P. Kleinschmidt. A constrained matching problem. *Annals of Operations Research*, 57(1):135–145, 1995.

[29] A. Itai, S. L. Tanimoto, and M. Rodeh. Some Matching Problems for bipartite Graphs. *Journal of the ACM (JACM)*, 25(4):517–525, 1978.

[30] D. Jungnickel. *Graphs, Networks and Algorithms.* Algorithms and Computation in Mathematics. Springer, Berlin, Heidelberg, New York, 2005.

[31] R. M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, The IBM Research Symposia Series, pages 85–103, New York, 1972. Plenum Press.

[32] A. V. Karzanov. Maximum matching of given weight in complete and complete bipartite graphs. *Cybernetics and Systems Analysis*, 23(1):8–13, 1987.

[33] G. Kochenberger, F. Glover, and B. Alidaee. An effective approach for solving the binary assignment problem with side constraints. *International Journal of Information Technology & Decision Making*, 1(1):121–129, 2002.

[34] S. O. Krumke and H. Noltemeier. *Graphentheoretische Konzepte und Algorithmen.* Teubner, Wiesbaden, 2005.

[35] H. W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97, 1955.

[36] H. W. Kuhn. Variants of the Hungarian method for assignment problems. *Naval Research Logistics Quarterly*, 3(4):253–258, 1956.

[37] E. L. Lawler. *Combinatorial Optimization: Networks and Matroids.* Holt, Rinehart and Winston, New York, 1976.

[38] J. K. Lenstra and A. H. G. R. Kan. Computational complexity of discrete optimization problems. *Annals of Discrete Mathematics*, 4:121–140, 1979.

[39] J. B. Mazzola and A. W. Neebe. Resource-constrained assignment scheduling. *Operations Research*, 34(4):560–572, 1986.

[40] J. B. Mazzola and A. W. Neebe. An algorithm for the bottleneck generalized assignment problem. *Computers and Operations Research*, 20(4):355–362, 1993.

[41] C. A. Meyers and A. S. Schulz. Integer equal flows. *Operations Research Letters*, 37(4):245–249, July 2009.

[42] D. Naddef. Rank of maximum matchings in a graph. *Mathematical Programming*, 22(1):52–70, 1982.

[43] G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization.* Wiley-Interscience, New York, NY, USA, 1988.

[44] T. Öncan. A survey of the generalized assignment problem and its applications. *INFOR: Information Systems and Operational Research*, 45(3):123–141, 2007.

[45] M. Padberg and A. Sassano. The complexity of matching with bonds. *Information Processing Letters*, 32(6):297–300, 1989.

[46] C. H. Papadimitriou and M. Yannakakis. The complexity of restricted spanning tree problems. *Journal of the ACM (JACM)*, 29(2):285–309, 1982.

[47] D. W. Pentico. Assignment problems: A golden anniversary survey. *European Journal of Operational Research*, 176(2):774–793, 2007.

[48] S. Sahni. Computationally Related Problems. *SIAM Journal on Computing*, 3(4):262–279, 1974.

[49] A. Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency.* Springer, Berlin, Heidelberg, 2003.

[50] K. Srinathan, P. R. Goundan, M. V. N. A. Kumar, R. Nandakumar, and C. P. Rangan. Theory of equal-flows in networks. In O. H. Ibarra and L. Zhang, editors, *Proceedings of the 8th Annual International Conference on Computing and Combinatorics*, Lecture Notes in Computer Science, pages 514–524. Springer, Berlin, Heidelberg, 2002.

[51] L. J. Stockmeyer and V. V. Vazirani. NP-completeness of some generalizations of the maximum matching problem. *Information Processing Letters*, 15(1):14–19, 1982.

[52] T. Yi, K. G. Murty, and C. Spera. Matchings in colored bipartite networks. *Discrete Applied Mathematics*, 121(1-3):261–277, 2002.