BRANCH-AND-PRICE ALGORITHMS FOR THE MAXIMUM WEIGHT PERFECT MATCHING PROBLEM WITH CONFLICT CONSTRAINTS

by

Alim Buğra Çınar

B.S., Industrial Engineering, Boğaziçi University, 2018

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Industrial Engineering

Boğaziçi University

2021

To Begüm

Her memory will be with me always.

ACKNOWLEDGEMENTS

Firstly, I would like to express my deepest gratitude to my advisor, İ. Kuban Altınel. He has always supported and guided me throughout this journey in the best possible way. It is a great honor and an excellent chance to work with you. I am also extremely grateful to my co-advisor Temel Öncan sincerely. His support and guidance are invaluable. Finally, I must thank Z. Caner Taşkın, Necati Aras, and Selda Küçükçiftçi, who are on my jury, for their evaluations and valuable advice.

I must express my sincere love to Özlem, the love of my life and my companion. You were always with me during the most enjoyable and the most difficult moments. It is impossible to put into words the value of your existence.

I gratefully thank my father, my mother, my sister Şule and her husband Sirer, my dear little ones Yağmur and Güneş, and especially my sister Sevinç, who is always by my side. I am really fortunate to have you as my family.

I would also like to thank Hilal, Okan, and my dear Deniz. The moments we spent together are invaluable.

I wish to thank all my colleagues, especially Tarkan, Elif, Selin, Feyyaz, Çiğdem, Orkun, Ekin, and Gizem, for their friendship and support. I will never forget the great memories we had together. I also gratefully acknowledge all the professors and staff in the Department of Industrial Engineering.

Finally, I would like to acknowledge the partial support of TÜBİTAK under grant no: 217M477 carried out by Temel Öncan within the 1001 program. I again want to thank TÜBİTAK for their financial support during my graduate study through the BİDEB-2210A scholarship.

ABSTRACT

BRANCH-AND-PRICE ALGORITHMS FOR THE MAXIMUM WEIGHT PERFECT MATCHING PROBLEM WITH CONFLICT CONSTRAINTS

In this thesis, we consider the Maximum Weight Perfect Matching Problem with Conflict Constraints (MWPMC), which is a generalization of the well-known the Maximum Weight Perfect Matching Problem (MWPMP). Although MWPMC is a relatively recent problem, there are applications modeled as MWPMC or using MWPMC as a subproblem in a wide spectrum from molecular biology to computer graphics. MWPMC is proven to be $\mathcal{NP}$-hard. Hence, a high-performance solution approach is crucial for both existing and potential applications.

There are a number of combinatorial optimization problems in the literature involving conflict constraints. We observe that some of the related studies had satisfactory results with Branch-and-Price (B&P) algorithm. Also, no studies aiming to solve MWPMC by using B&P exist to the best of our knowledge. Therefore, we study the question of how B&P algorithm can be applied to the MWPMC. We propose several Integer Programming (IP) formulations for the MWPMC and build corresponding B&P schemes. Then, B&P algorithms are implemented and tested. According to our findings, some of the proposed schemes show performance that can compete with the commercial solvers.

ÖZET

ÇATIŞMA KISITLI EN BÜYÜK AĞIRLIKLI TAM EŞLEME PROBLEMİ İÇİN DAL-FİYAT ALGORİTMALARI

Bu çalışmada En Büyük Ağırlıklı Tam Eşleme Probleminin genel bir hali olan Çatışma Kısıtlı En Büyük Ağırlıklı Tam Eşleme Problemini (ÇETEP) ele alıyoruz. Görece yeni bir problem olmasına karşın, moleküler biyolojiden bilgisayarlı çizime kadar birçok alanda ÇETEP olarak gösterimlenen ya da ÇETEP'i altproblem olarak kullanan uygulamalar bulunmaktadır. ÇETEP, $\mathcal{NP}$-zor olduğu kanıtlanmış bir problemdir. Dolayısıyla, ÇETEP için etkin bir çözüm yaklaşımı geliştirmek, hem var olan hem de olası uygulamalar için çok önemlidir.

Bilimsel yazında, çatışma kısıtları ile tanımlanmış bir dizi birleşimsel eniyileme problemi bulumaktadır. Bu çalışmaların bir kısmının Dal-Fiyat (DF) algoritmasını ile tatminkar sonuçlar verdiğini görüyoruz. Ayrıca, güncel bilgilerimize göre ÇETEP'i DF yardımıyla çözmeyi amaçlayan bir çalışma bulunmamaktadır. Bu nedenle, DF yaklaşımının ÇETEP'e nasıl uygulanabileceği sorusuna yanıt aramanın ilginç olacağını düşündük. ÇETEP için birkaç Tamsayı Programlama (TP) gösterimi önerdik, DF algoritmaları geliştirdik ve bunları programlayarak sınadık. Bulgularımıza göre bunlardan bazıları ticari çözücüler ile rekabet edebilecek başarımdadır.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

$\mathbf{A}_G$	The adjacency matrix of graph G
$\underline{\mathbf{A}}_G$	The cardinality adjacency matrix of graph G
$\mathbf{B}_G$	The incidence matrix of graph G
C	A conflict graph
$\hat{C}$	An extended conflict graph
$\mathbf{d}_G$	The degree vector of the vertices of graph G
$E(G)$	The edges of a graph G
F	Free Set
G	A general graph
$\bar{G}$	The complement of a graph G
$G[\![F]\!]$	The edge-induced subgraph induced by edges $F \subseteq E(G)$
$G[W]$	The vertex-induced subgraph induced by vertices $W \subseteq V(G)$
$\{G_1, \ldots, G_P, \tilde{G}\}$	A partitioning of graph G
$\mathcal{G}$	A generating set
I	Included Set
$\bar{\mathbf{I}}_n$	A vector of length n where all elements are equal to I
K	A clique
$\mathcal{K}_G$	The set of maximal cliques of graph G
$\mathbf{K}_G$	The clique matrix of graph G
$L(G)$	The line graph of graph G
M	A matching
$N_G(v)$	The neighborhood of vertex v in graph G
S	A stable-set
$\|s\|$	The cardinality of an arbitrary set s
$V(G)$	The vertices of a graph G
X	Excluded Set
$\delta_G(v)$	The set of edges of G that are incident with vertex v

Γ	Graph Density
$\gamma_G(W)$	The set of edges of G that has both ends in the vertex set W
Ω	Conflict Graph Density

LIST OF ACRONYMS/ABBREVIATIONS

APC	Assignment Problem with Conflicts
BIP	Binary Integer Programming
BPC	Bin Packing Problem with Conflicts
B&B	Branch-and-Bound
B&P	Branch-and-Price
CG	Column Generation
DCKP	Disjunctively Constrained Knapsack Problem
IP	Integer Programming
IPM	Integer Programming Master
KPC	Knapsack Problem with Conflicts
LP	Linear Programming
LPM	Linear Programming Master
MCFPC	Minimum Cost Flow Problem with Conflicts
MCNFP	Minimum Cost Noncrossing Flow Problem on Layered Networks
MCP	Maximum Clique Problem
MFPC	Maximum Flow Problem with Conflicts
MIP	Mixed Integer Programming
MMP	Maximum Matching Problem
MSSP	Maximum Stable Set Problem
MWCC	Maximum Weight Clique Problem with Cardinality Constraints
MWCP	Maximum Weight Clique Problem
MWMP	Maximum Weight Matching Problem
MWPMC	Maximum Weight Perfect Matching Problem with Conflict Constraints
MWPMP	Maximum Weight Perfect Matching Problem

MWSSC	Maximum Weight Stable Set Problem with Cardinality Constraints
MWSSP	Maximum Weight Stable Set Problem
RLPM	Restricted Linear Programming Master
SP	Subproblem

1. INTRODUCTION

The matching theory is an important and well-studied area of combinatorial optimization. A broad range of real-world applications such as personnel assignment, dual completion of oil wells, locating objects in space, optimal depletion of inventory, and scheduling on parallel machines can be formulated as matching problems [1]. Combinatorial, graph theoretical, and linear programming foundations of the matching theory are presented in [2].

We address the *Maximum Weight Perfect Matching Problem with Conflict Constraints* (MWPMC) in this thesis. The classical *Maximum Weight Perfect Matching Problem* (MWPMP) is extended with the conflict constraints to acquire MWPMC. The conflict constraints are also named as disjunctive constraints or exclusionary side constraints in the literature [3].

A matching can be defined as a set of pairs in a group of objects. If the matching covers all objects, it is called a perfect matching. Finding a perfect matching with the maximum weight, where the weight of the matching is the sum of the weights associated with the pairs in it, is the aim of MWPMP. The conflict constraints restrict some of the pairs to be in a matching at the same time. Hence, a solution to MWPMP may not be a solution to the MWPMC in case it contains conflicting pairs. While MWPMP is easy to solve, solving MWPMC exactly is intractable.

Branch-and-Price (B&P) is a technique that is used to solve very large scale Integer Programming (IP) problems [4]. However, any IP regardless of its number of variables can be solved by B&P, after transforming it into an appropriate form that contains a very large number of variables. The relaxation of the transformed IP problem is solved in each iteration of B&P by Column Generation (CG) method, which is an approach for solving Linear Programming (LP) problems with a very large number of variables. For a complete overview of the underlying theory we refer to [5].

The remainder of the thesis is organized as follows. A short background on the basic graph terminology is given in Chapter 2 in order to increase readability. In Chapter 3 the literature on the subject is reviewed. Then, the problem is formally defined in Chapter 4 and several IP formulations are proposed. In Chapter 5, the details on the construction of the B&P schemes are explained. The algorithms are discussed in Chapter 6 and the results are analyzed in Chapter 7. The thesis is concluded with Chapter 8.

2. BACKGROUND

In this chapter, we try to build a background that seems helpful while reading the rest of the thesis. First we introduce basic graph definitions and structures. Then, the matrix representations that are frequently used throughout the work are presented. After introducing conflict graphs, we define the common graph problems that are closely related to MWPMC. A combination of the notations that take place in [6–9] are employed in this thesis.

2.1. Graph Definitions and Notation

A *graph* $G = (V(G), E(G))$ is composed of a set of *vertices* $V(G)$ and a set of *edges* $E(G)$ between the vertices. In this thesis, we only consider finite graphs which has no parallel edges and no loops, i.e., finite simple graphs. *Complement* of a graph G, which is denoted as $\bar{G} = (V(\bar{G}), E(\bar{G}))$, is defined by $V(\bar{G}) = V(G)$ and $E(\bar{G}) = \{\{u, v\} \notin E(G) : u, v \in V(G)\}$. Let $N_G(v)$, which is called the *neighborhood of* v, be the set of vertices adjacent to vertex v but not containing v in G. Let $\delta_G(v)$ represent the set of edges incident with vertex v in G. We let $\gamma_G(W)$ denote the set of edges of G with both ends in $W \subseteq V(G)$. Finally, $\mathbf{d}_G(v) = |N_G(v)|$ stands for the *degree* of v in G, which is the number of vertices adjacent to it.

A *vertex-induced graph* and an *edge-induced graph* are defined by subsets $W \in V(G)$ and $F \in E(G)$, respectively. A subgraph of G induced by the vertex subset W is denoted by $G[W] = (W, \gamma_G(W))$, while $G[\![F]\!] = (V_F, F)$ where $V_F = \{v \in V(G) : \delta_G(v) \cap F \neq \varnothing\}$ is the induced subgraph of G by the edge subset F. Note that, $\gamma_G(V_F) = F$ holds. We define a *partitioning* of a graph G by the set $\{G_1, \ldots, G_P, \tilde{G}\}$ which contains $P+1$ subgraphs. Subgraphs $G_i = G[V(G_i)]$ are defined by P distinct partitions of $V(G) = \bigcup_{i=1}^{P} V(G_i)$. The edges in G that have ends between different vertex partitions construct an edge-induced subgraph $\tilde{G} = (G[\![\tilde{E}]\!] : \tilde{E} = E(G) / \bigcup_{i=1}^{P} \gamma_G(V(G_i))$.

A *matching* M in G is a subset of edges $E(G)$ which are not incident to a common vertex. In other words, a vertex is incident with at most one edge in a matching. The size of a matching is the number of edges in it. If the size of a matching is equal to half of the number of vertices, then it is called a *perfect matching*, which contains an edge incident to each vertex. Hence, a perfect matching could exist only in graphs with an even number of vertices. In an edge-weighted graph, the weight of a matching is the sum of the weights of the edges in the matching.

A *stable set* S, which is also called an independent set or a vertex packing, is a set of vertices that contains no adjacent pair. A *clique* K is a complete subgraph of a graph, where all vertices are adjacent to each other. Cliques and stable sets are closely related. A stable set in graph G forms a clique in the complement graph $\bar{G}$ and vice versa. In a vertex-weighted setting, the sum of the weights of vertices in a stable set (clique) is the weight of the stable set (clique).

A clique, where it is not possible to add another vertex without violating clique property, is named as a *maximal clique*. A maximal clique's subsets are also cliques. Nevertheless, the maximal clique itself cannot be a proper subset of any other clique. A graph can contain more than one maximal clique. The largest maximal clique is called the *maximum clique*. We denote the set of maximum cliques in a graph G as $\mathcal{K}_G$.

A *maximal stable set* is defined as a stable set in which it is not possible to add one more vertex without violating the stable set property. Hence, a maximal stable set is not a proper subset of any other stable set in the graph. The stable set of largest size is called the *maximum stable set*, which is also a maximal stable set. On the other hand, all maximal stable sets are not necessarily maximum stable sets.

2.2. Matrix Representations

Matrices can be used to represent graphs. An *adjacency matrix* $\mathbf{A}_G$ is a $|V(G)| \times |V(G)|$ binary matrix whose ij^{th} element equals to one if and only if vertices i and j are adjacent in G. The *cardinality adjacency matrix* $\underline{\mathbf{A}}_G$ is built by inserting the degrees of the vertices to the diagonal elements of $\mathbf{A}_G$, namely, ii^{th} element is equal to $d_G(i)$. The *incidence matrix* $\mathbf{B}_G \in \mathbb{B}^{|V(G)| \times |E(G)|}$ represents vertex-edge incidence relationships, where ij^{th} element equals to one if and only if edge j is incident with vertex i in G. Finally, a *clique matrix* $\mathbf{K}_G$ is also a binary matrix with dimensions $|\mathcal{K}_G| \times |E(G)|$, which contains a row for each maximal clique and has ij^{th} element equals to one if and only if the i^{th} clique contains edge j in G. We illustrate these matrices for the particular graph given in the following figure.

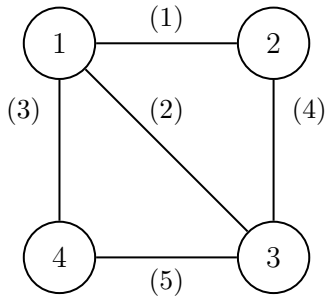

Figure 2.1. A example for G.

The adjacency and the cardinality adjacency matrices of G are given as:

$\mathbf{A}_G$	1	2	3	4
1	0	1	1	1
2	1	0	1	0
3	1	1	0	1
4	1	0	1	0

Figure 2.2. Adjacency matrix.

$\underline{\mathbf{A}}_G$	1	2	3	4
1	3	1	1	1
2	1	2	1	0
3	1	1	3	1
4	1	0	1	2

Figure 2.3. Cardinality adjacency matrix.

Observe that all entries except the diagonal entries are the same in both matrices. Also, the diagonal entries are the degrees of the corresponding vertices in the cardinality adjacency matrix.

Figure 2.4 shows the incidence matrix of G where the columns correspond to the edges, while rows represent the vertices. Observe that only the cells corresponding to the edges in $\delta(v)$ have value one in the v^{th} row in $\mathbf{B}_G$. Also, each column has only two entries having value of one, which correspond to the ends of the edge that the column represents. The clique matrix of G is depicted with Figure 2.5, which has two rows corresponding to both maximal cliques G, i.e. $\{1, 2, 3\}$ and $\{1, 3, 4\}$. Note that, the columns represents the edges of G in the clique matrix. Hence, the ij^{th} entry is equal to one when edge j is contained in maximal clique i.

$\mathbf{B}_G$	1	2	3	4	5
1	1	1	1	0	0
2	1	0	0	1	0
3	0	1	0	1	1
4	0	0	1	0	1

Figure 2.4. Incidence matrix.

$\mathbf{K}_G$	1	2	3	4	5
1	1	1	0	1	0
2	0	1	1	0	1

Figure 2.5. Clique matrix.

2.3. The Conflict Graph

We mentioned the function of conflict constraints as restricting some edge pairs in a matching at the same time. The pairs in a matching are shown by the edges. Therefore, a conflict constraint simply restricts two conflicting edges of graph G to be in a matching at the same time. We represent this relationship by a *conflict graph* $C = (V(C), E(C))$. $V(C) \equiv E(G)$ and there exists an edge between the vertices u and v in C if the edges u and v in G conflict. Hence, $E(C)$ represents the conflicts among the edges in $E(G)$.

A *line graph* $L(G) = (V(L(G)), E(L(G)))$ of G contains a vertex for each edge in G, similar to the conflict graph C. If the edges u and v in G are incident with a common vertex, then $\{u, v\} \in E(L(G))$. The edges of G translates into vertices in $L(G)$ while adjacency relationships among them are represented by $E(L(G))$. A matching is defined as an independent edge set. Hence, there is a one-to-one correspondence between matchings in G and stable sets in $L(G)$.

Let edges 1 and 5 be in conflict in the sample graph G shown in Figure 2.1. Then, the corresponding conflict graph C is depicted with Figure 2.6, which contains a single edge $\{1, 5\}$ while it has a vertex for each vertex of G. Figure 2.7 illustrates the line graph $L(G)$, which also has the same vertex set, while it has an edge for each adjacent edge pair in G.

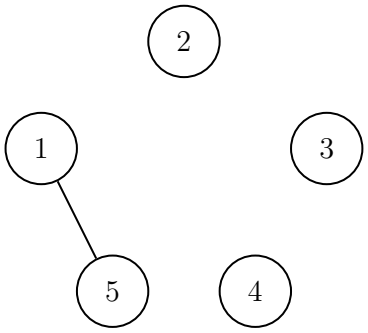

Figure 2.6. Conflict graph C.

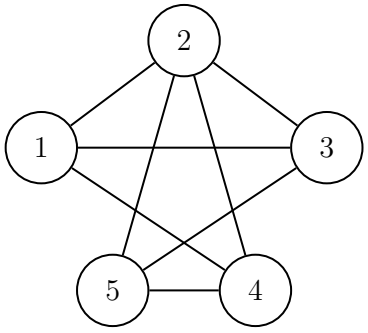

Figure 2.7. Line graph $L(G)$.

Both $L(G)$ and C of a general graph G contains the same set of vertices while the edge sets are different. We can define another graph called the *extended conflict graph* $\hat{C}$ by joining C and $L(G)$. $V(\hat{C}) \equiv E(G)$ and $E(\hat{C}) = E(C) \cup E(L(G))$. Hence, $\hat{C}$ contains information regarding both adjacency and conflict relationships among the edges of G. The corresponding extended conflict graph of the sample graph is illustrated with Figure 2.8.

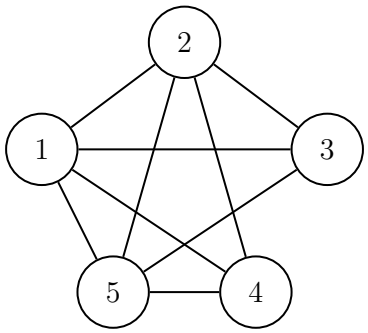

Figure 2.8. Extended conflict graph.

2.4. Graph Problems

The *Maximum Matching Problem* (MMP) aims to find the matching of maximum size in a graph. The problem of finding a matching with the largest weight is the *Maximum Weight Matching Problem* (MWMP). When all weights are equal to one in MWMP, the special case MMP is acquired. If the matching is also required to be perfect, the problem turns into *Maximum Weight Perfect Matching Problem* (MWPMP). These problems can be solved in polynomial time [2].

The problem of finding a stable set of maximum size is the *Maximum Stable Set Problem* (MSSP), which is $\mathcal{NP}$-hard [10]. In a vertex-weighted graph, the weight of a stable set is the sum of the weights of the vertices in the set. The *Maximum Weight Stable Set Problem* (MWSSP) deals with the problem of finding a stable set with largest possible weight. The weighted case is also $\mathcal{NP}$-hard, as it is a generalization of MSSP.

There is a complementary relationship between the cliques and the stable sets. The maximum stable set in G forms a clique in $\bar{G}$ which is the maximum clique. The reverse is also true. Hence, solving MSSP in G is equivalent to solving the Maximum Clique Problem (MCP). The relationship also holds for the weighted versions, MWSSP in G and Maximum Weight Clique Problem (MWCP) in $\bar{G}$. Both clique problems, MCP and MWCP, are $\mathcal{NP}$-hard.

The clique and stable set problems in the line graphs can be solved in polynomial time [2]. A stable set (clique) in $L(G)$ ($\overline{L(G)}$) is equivalent to an independent edge set in G and vice versa. Therefore, to solve MWSSP in $L(G)$ (MWCP in $\overline{L(G)}$) we can firstly solve MWMP in G, then transform the solution in polynomial time [2]. A similar relationship can be constructed between MWPMC and the Maximum Weight Stable Set (Clique) Problem with Cardinality Constraints (MWSSC (MWCC)), where the cardinality is restricted to be $\frac{|V(G)|}{2}$. To solve MWSSC (MWCC) in $L(G)$ ($\overline{L(G)}$), we first solve MWPMP in G, then transform the solution. Hence, we can solve both MWSSC and MWCC in polynomial time.

3. LITERATURE REVIEW

3.1. Matching Problems with Conflict Constraints

The earliest work that we are aware of on a matching problem with conflict constraints is [3], where extensions of minimum spanning tree, maximum matching, and shortest path problems with two types of binary disjunctive constraints, negative and positive disjunctions, were studied. Both MMC and MWMC are shown to be $\mathcal{NP}$-hard, even in the case that the conflict graph is a set of independent edges.

Additional complexity results including two polynomially solvable cases of MWPMC were shown in [11]. They also drew attention to the fact that it is not possible to generate polynomial-time cases by restricting the conflict graph, following the result in [3] on the structure of the conflict graph. An IP formulation and a quadratic formulation were presented. They also proposed five heuristics and two lower bounding approaches for the MWPMC on complete bipartite graphs, which is the Assignment Problem with Conflicts (APC). Another work on APC, which proposes an exact B&B algorithm, heuristics, and branching rules is [12]. Finally, a B&B algorithm and a Russian Doll Search algorithm for APC were discussed in [13].

The only exact algorithm for the MWPMC that we are aware of is proposed in [14]. It is a B&B algorithm using MWMC relaxation for upper bounding. Also, there exist applications of matching problems with conflict constraints. For instance, the RNA-folding problem, which is an application in the computational and systems biology, is formulated as MWMC in [15]. Another application is in computer graphics and geometry processing. A method for the problem of automatic generation of the patch layouts of complex geometrical objects, which includes solving MWPMC as a subproblem, was proposed in [16]. The authors emphasized that the performance of the MWPMC solver is crucial for the overall performance of the generator.

3.2. Conflict Constraints in General

The notion of conflicts is applied to many combinatorial optimization problems. We already mentioned that minimum spanning tree and shortest path problems with conflict constraints were studied in [3] and assignment problem with conflicts in [13]. There are also studies focusing on the conflict constrained extensions of Knapsack, Bin Packing, Max Flow, and Min Cost Flow problems in the literature. Moreover, there are efficient B&P applications to some of these problems.

To the best of our knowledge, the Knapsack Problem with Conflicts (KPC), or Disjunctively Constrained Knapsack Problem (DCKP), was firstly studied in [17], where an IP formulation and a B&B algorithm were proposed. Also, a greedy heuristic, a local search heuristic, several pruning rules, and an interval reduction approach were discussed. A heuristic algorithm for KPC and another B&B algorithm were proposed in [18] and [19] respectively. Pseudo-polynomial algorithms for KPC on two graph classes and approximation schemes were studied in [20]. A more recent study considering KPC is [21], which proposes a B&B algorithm that outperforms a state-of-the-art commercial solver on instances having 10% or larger density. Finally, the polytope of KPC, new sets of valid inequalities and a branch-and-cut algorithm utilizing the proposed results were proposed in [22]. They also shown that the proposed algorithm outperforms a general-purpose solver on hard instances.

One of the earliest studies, where a polynomial-time approximation scheme is developed, considering Bin Packing with Conflicts (BPC) problem is [23]. Heuristic approaches and lower bounding schemes for BPC were proposed in [24]. More recent studies, which rely on B&P algorithm to find an exact solution, on BPC are [25–27]. They all use similar formulations, where KPC appears as a subproblem. A greedy heuristic to find a KPC solution and a branching strategy based on the master problem variables was proposed in [25]. A branching method based on the Ryan-Foster rule [28] was offered in [26]. Finally, a dynamic programming approach and a depth-first-search B&B algorithm for solving the subproblem KPC were studied in [27].

The maximum flow problem subject to the binary disjunctive constraints is firstly addressed in [29]. The negative disjunctive constraints correspond to the conflict relationships. Even if a conflict graph only consists of independent edges, the maximum flow problem with conflicts (MFPC) is shown to be $\mathcal{NP}$-hard. The study also shows that no polynomial-time approximation algorithm exists for the MFPC. Three exact algorithms, which are Benders decomposition, B&B, and Russian Doll Search, to solve the MFPC were proposed in [30].

The minimum cost noncrossing flow problem on layered networks (MCNFP) is studied in [31]. The complexity of the problem is proven to be $\mathcal{NP}$-hard. Mixed Integer Programming (MIP) formulations and polynomial cases for the MCNFP and a preprocessing algorithm are proposed. The generalization of MCNFP, which is the minimum cost flow problem with conflicts (MCFPC), is studied in [32]. In this study, a B&B algorithm and a Russian Doll Search algorithm are proposed for solving the MCFPC exactly, in addition to preprocessing rules increasing the efficiency of the algorithms.

4. COMPLEXITY AND FORMULATION

In this chapter, several IP formulations is proposed for both MWPMC and MWSSC. Also, the equivalence between MWPMC and MWSSC on the extended conflict graph is discussed. Both of these problems are known to be $\mathcal{NP}$-hard. A trivial proof for the MWPMC is shown in Proposition 4.1. The result for MWSSC on $\hat{C}$ also similarly follows.

Proposition 4.1. *MWPMC is $\mathcal{NP}$-hard.*

Proof. We know that optimization version of a problem can be solved in polynomial time if and only if the decision version can be answered in polynomial time. MMC is known to be $\mathcal{NP}$-hard. Hence, the decision version of MMC is $\mathcal{NP}$-complete. Let us define the decision versions of both MMC and MWPMC as follows.

MMC (D)
INSTANCE: A graph G, a conflict graph C, an integer k.
QUESTION: Is there a conflict-free matching in G of size $\geq k$?

MWPMC (D)
INSTANCE: A graph G, a conflict graph C, edge weights w.
QUESTION: Is there a conflict-free perfect matching in G of weight $\geq W$?

When all weights w equal to one, MWPMC (D) reduces to MMC (D) with $W = k = \frac{|V(G)|}{2}$. Thus, MWPMC (D) is $\mathcal{NP}$-complete and MWPMC is $\mathcal{NP}$-hard. □

Recall that $\mathbf{B}_G$ denotes the incidence matrix of G. Let $\bar{\mathbf{1}}_{|V(G)|}$ be a vector of length $|V(G)|$, where all entries are equal to 1. Finally, $\mathbf{w}$ corresponds to the weights of the edges in a graph G.

An IP formulation for MWPMC is given as

$$\begin{aligned}
\text{MWPMC:} \quad \max \quad & \mathbf{w}^T\mathbf{x} && (4.1)\\
\text{s.t.} \quad & \mathbf{B}_G\mathbf{x} = \bar{\mathbf{1}}_{|V(G)|} && (4.2)\\
& \mathbf{B}_C^T\mathbf{x} \leq \bar{\mathbf{1}}_{|E(C)|} && (4.3)\\
& \mathbf{x} \in \mathbb{B}^{|E(G)|}, && (4.4)
\end{aligned}$$

where each binary decision variable corresponds to an edge in the original graph G. The objective function (4.1), which is maximized, is the total weight. The matching constraint (4.2) enforces the sum of the variables corresponding to the edges incident to each vertex v to be equal to one. We call (4.3) the conflict constraint, which enforces the sum of the variables corresponding to the edges in each conflicting pair to be at most one.

There is a one-to-one correspondence between matchings in a graph G and stable-sets in the line graph $L(G)$. Hence, the maximum weight conflict-free perfect matching in G is the maximum weight conflict-free stable-set of size $\frac{|V(G)|}{2}$ in $L(G)$. The extended conflict graph $\hat{C}$ is constructed by joining C and $L(G)$. Hence, MWPMC in G and MWSSC in $\hat{C}$ are equivalent.

An equivalent MWSSC formulation is obtained by replacing matching constraints (4.2) with a cardinality constraint that enforces the sum of the edges in a feasible solution to be equal to the half of the number of vertices. Also, defining conflict constraints for each edge in the extended conflict graph is required. After applying these changes, MWSSC formulation becomes

$$\begin{aligned}
\text{MWSSC:} \quad \max \quad & \mathbf{w}^T\mathbf{x} && (4.5)\\
\text{s.t.} \quad & \bar{\mathbf{1}}_{|E(G)|}^T\mathbf{x} = \frac{|V(G)|}{2} && (4.6)\\
& \mathbf{B}_{\hat{C}}^T\mathbf{x} \leq \bar{\mathbf{1}}_{|E(\hat{C})|} && (4.7)\\
& \mathbf{x} \in \mathbb{B}^{|E(G)|}. && (4.8)
\end{aligned}$$

We call both MWPMC and MWSSC models as the strong formulations concerning the representation of conflicts. It is possible to obtain equivalent weak and clique formulations. The conflict constraints in the weak formulation of MWPMC are formed by aggregating the conflict inequalities represented with (4.3) for each vertex $v \in V(C)$ over the edges of its neighbors $N_C(v)$. The results are represented by the cardinality adjacency matrix ($\underline{\mathbf{A}}_C$), which is constructed by adding the i^{th} entry of the degree vector of the vertices of C ($\mathbf{d}_C(i)$) to the ii^{th} entry of the adjacency matrix of C, on the left-hand side. The right-hand side vector becomes $\mathbf{d}_C$. The resulting weak conflict constraint for MWPMC is

$$\underline{\mathbf{A}}_C \mathbf{x} \leq \mathbf{d}_C. \tag{4.9}$$

We construct the weak conflict constraints for MWSSC by following the same procedure, whose result can be obtained by simply replacing C with $\hat{C}$ in (4.9).

The clique formulation is defined for the set of maximal cliques in the (extended) conflict graph. Each constraint corresponds to a maximal clique and enforces the number of vertices selected from each maximal clique to be at most one.

$$\mathbf{K}_C \mathbf{x} \leq \bar{\mathbf{1}}_{|\mathcal{K}_C|} \tag{4.10}$$

is the clique constraint, which replaces (4.3) in the clique formulation. The left-hand side of the equation contains the clique matrix of C ($\mathbf{K}_C$), which contains a row for each maximal clique in C and a column for each edge. The clique conflict constraints for MWSSC can be constructed by replacing C with $\hat{C}$ in both sides of the equation.

In addition to the strong, weak and clique formulations, we propose a formulation based on the partitioning of the conflict graph ($\{C_1, \ldots, C_P, \tilde{C}\}$) as

$$\text{MWPMC Partition:} \quad \max \quad \sum_{p=1}^{P} \mathbf{w}_p^T \mathbf{x}_p \tag{4.11}$$

$$\text{s.t.} \quad \sum_{p=1}^{P} \mathbf{B}_G(p)\mathbf{x}_p = \bar{\mathbf{1}}_{|V(G)|} \tag{4.12}$$

$$\sum_{p=1}^{P} \mathbf{B}_{\tilde{C}}^T(p)\mathbf{x}_p \leq \bar{\mathbf{1}}_{|E(\tilde{C})|} \tag{4.13}$$

$$\mathbf{B}_{C_p}^T \mathbf{x}_p \leq \bar{\mathbf{1}}_{|E(C_p)|} \qquad p = 1, 2, \ldots, P \tag{4.14}$$

$$\mathbf{x}_p \in \mathbb{B}^{|V(C_p)|} \qquad p = 1, 2, \ldots, P. \tag{4.15}$$

The idea behind partitioning formulations is due to [6]. The conflict constraints are formulated with respect to all the subgraphs in a partitioning. As all the edges in conflict graph are represented either in one of the vertex-induced subgraphs or in the edge-induced subgraph in the partitioning, the constraints (4.13) and (4.14) covers all the edges. Model (4.11)-(4.15) is the partitioning formulation based on (4.1)-(4.4). Note that $\mathbf{B}_G(p)$ and $\mathbf{B}_{\tilde{C}}^T(p)$ are binary matrices of sizes $|V(G)| \times |V(C_p)|$ and $|E(\tilde{C})| \times |V(C_p)|$. They only contain the columns of $\mathbf{B}_G$ and $\mathbf{B}_{\tilde{C}}^T$ corresponding to $V(C_p)$.

It is also possible to construct a partitioning formulation for MWSSC based on the partitioning $\{\hat{C}_1, \ldots, \hat{C}_P, \tilde{\hat{C}}\}$ of the extended conflict graph. In the partitioning formulation of MWSSC, which is modeled as

$$\text{MWSSC Partition:} \quad \max \quad \sum_{p=1}^{P} \mathbf{w}_p^T \mathbf{x}_p \tag{4.16}$$

$$\text{s.t.} \quad \sum_{p=1}^{P} \bar{\mathbf{1}}_{|V(\hat{C}_p)|}^T \mathbf{x}_p = \frac{|V(G)|}{2} \tag{4.17}$$

$$\sum_{p=1}^{P} \mathbf{B}_{\tilde{\hat{C}}}^T(p)\mathbf{x}_p \leq \bar{\mathbf{1}}_{|E(\tilde{\hat{C}})|} \tag{4.18}$$

$$\mathbf{B}_{\hat{C}_p}^T \mathbf{x}_p \leq \bar{\mathbf{1}}_{|E(\hat{C}_p)|} \qquad p = 1, 2, \ldots, P \tag{4.19}$$

$$\mathbf{x}_p \in \mathbb{B}^{|V(C_p)|} \qquad p = 1, 2, \ldots, P. \tag{4.20}$$

In this formulation, (4.7) is replaced by (4.18) and (4.19) considering the partitioning. Also, (4.16) and (4.17) are redefined by using the vectors that are defined considering $V(C_p)$ in (4.20). $\mathbf{B}^T_{\hat{\tilde{C}}}(p)$ notation is also used in this model considering the columns representing $V(\hat{C}_p)$.

Both MWPMC Partition and MWSSC Partition models are constructed based on the strong formulations. To obtain the weak partition formulations, all conflict constraints must be reformulated with cardinality adjacency matrices and the degree vectors corresponding to the subgraphs that they are defined on. The resulting weak conflict constraints have the form shown in Equation (4.9). Similarly, the clique matrices of the subgraphs must be used to construct a clique partition formulation, where the conflict constraints are formulated as in Equation (4.10). One should notice that the weak partition formulations are stronger than the weak formulations. The reason is that the aggregation is done on smaller subgraphs, which produces stronger constraints. On the other hand, the clique partition formulations are weaker than the original clique formulations as the cliques on the subgraphs are smaller than the cliques on the original graph.

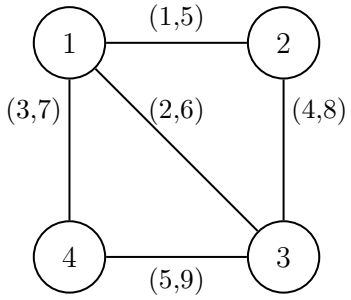

Figure 4.1. Weighted graph.

Now, let us illustrate the formulations through a set of small examples. We use the graph given in Figure 4.1, which is the weighted version of the graph in Figure 2.1, to build the formulations, where the labels over the edges show the id and weight of the edge as (id,weight). Let edges 1 and 5 be in conflict. Then, the strong MWPMC formulation for this graph is

$$\begin{aligned}
\text{MWPMC:} \quad \max \quad & 5x_1 + 6x_2 + 7x_3 + 8x_4 + 9x_5 && (4.21)\\
\text{s.t.} \quad & x_1 + x_2 + x_3 = 1 && (4.22)\\
& x_1 + x_4 = 1 && (4.23)\\
& x_2 + x_4 + x_5 = 1 && (4.24)\\
& x_3 + x_5 = 1 && (4.25)\\
& x_1 + x_5 \leq 1 && (4.26)\\
& x_1, x_2, x_3, x_4, x_5 \in \{0, 1\}, && (4.27)
\end{aligned}$$

where the matching constraints are (4.22)-(4.25) and the conflict constraint is (4.26). As there is a single conflict constraint, the weak and the clique formulations are also the same. Moreover, the partitioning formulation is the same for this example, if we partition the conflict graph shown in Figure 2.6 such that the single edge is contained in a partition. Similarly, the extended conflict graph that is given in Figure 2.8 can be used to formulate an equivalent MWSSC as

$$\begin{aligned}
\text{MWSSC:} \quad \max \quad & 5x_1 + 6x_2 + 7x_3 + 8x_4 + 9x_5 && (4.28)\\
\text{s.t.} \quad & x_1 + x_2 + x_3 + x_4 + x_5 = 2 && (4.29)\\
& x_1 + x_2 \leq 1 && (4.30)\\
& x_1 + x_3 \leq 1 && (4.31)\\
& x_1 + x_4 \leq 1 && (4.32)\\
& x_1 + x_5 \leq 1 && (4.33)\\
& x_2 + x_3 \leq 1 && (4.34)\\
& x_2 + x_4 \leq 1 && (4.35)\\
& x_2 + x_5 \leq 1 && (4.36)\\
& x_3 + x_5 \leq 1 && (4.37)\\
& x_4 + x_5 \leq 1 && (4.38)\\
& x_1, x_2, x_3, x_4, x_5 \in \{0, 1\}. && (4.39)
\end{aligned}$$

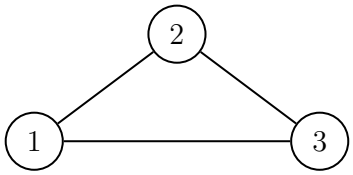

Figure 4.2. Partition 1.

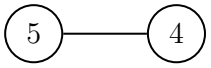

Figure 4.3. Partition 2.

We can partition the edges in the extended conflict graph as $V_1 = \{1, 2, 3\}$ and $V_2 = \{4, 5\}$. Figure 4.2 and Figure 4.3 illustrate the vertex-induced subgraphs of the extended conflict graph in Figure 2.8, which are constructed by the vertex sets V_1 and V_2, respectively. Figure 4.4 is the edge-induced subgraph that is built by the edges of the extended conflict graph that are not contained in both of the vertex-induced graphs. It is possible to obtain a partitioning formulation of MWSSC by using the illustrated subgraphs. Note that, when the strong formulation is used to model the conflict relationships in the partitioning formulation, only the conflict constraints (4.30)-(4.38) are permuted with respect to the edges of the subgraphs.

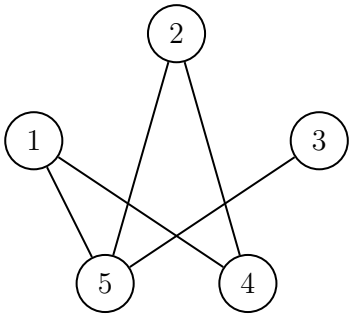

Figure 4.4. Edge-induced subgraph.

5. DANTZIG-WOLFE REFORMULATIONS

In this chapter, we propose five column generation schemes. Three of them contain a single subproblem, which are the perfect-matching-based and the stable-set-based formulations of MWPMC and the stable-set based formulation of MWSSC. The remaining two are based on the partitioning of MWPMC and MWSSC and contain multiple subproblems.

5.1. Formulations with a Single Subproblem

5.1.1. The Perfect-Matching-Based Formulation

Let us define a *generating set* $\mathcal{G}$, which contains all perfect matchings in graph G as

$$\mathcal{G} := \{\bar{\mathbf{x}} \in \mathbb{B}^{|E(G)|} : \mathbf{B}_G \bar{\mathbf{x}} = \bar{\mathbf{1}}_{|V(G)|}\}, \tag{5.1}$$

where each item $\bar{\mathbf{x}}_i$ in $\mathcal{G}$ is a binary vector of length $|E(G)|$. The j^{th} element $\bar{\mathbf{x}}_{ij}$ is equal to one, if edge $j \in E(G)$ belongs to the perfect matching represented by $\bar{\mathbf{x}}_i$. $\mathcal{G}$ can have an exponential number of items due to the number of perfect matchings in a graph.

It is possible to define the decision variables (4.4) with the items in $\mathcal{G}$ by the following equations

$$\begin{aligned} \mathbf{x} &= \sum_{k=1}^{|\mathcal{G}|} \bar{\mathbf{x}}_k \lambda_k \\ \sum_{k=1}^{|\mathcal{G}|} \lambda_k &= 1 \\ \lambda_k &= \{0,1\} \quad k = 1, 2, \ldots, |\mathcal{G}|. \end{aligned} \tag{5.2}$$

After redefining variables (4.4), the resulting model is an IP model with a large number of variables and called the Integer Programming Master (IPM) that is given as

$$\text{IPM:} \quad \max \quad \sum_{k=1}^{|\mathcal{G}|} (\mathbf{w}^T \bar{\mathbf{x}}_k) \lambda_k \tag{5.3}$$

$$\text{s.t.} \quad \sum_{k=1}^{|\mathcal{G}|} (\mathbf{B}_C^T \bar{\mathbf{x}}_k) \lambda_k \leq \bar{\mathbf{1}}_{|E(C)|} \tag{5.4}$$

$$\bar{\mathbf{1}}_{|\mathcal{G}|}^T \boldsymbol{\lambda} = 1 \tag{5.5}$$

$$\boldsymbol{\lambda} \in \mathbb{B}^{|\mathcal{G}|}. \tag{5.6}$$

The decision variable $\boldsymbol{\lambda}$ is a binary vector of size $|\mathcal{G}|$, which represent the coefficients of the items in $\mathcal{G}$ in (5.2). In the objective function (5.3), the coefficient of λ_k is the weight of $\bar{\mathbf{x}}_k \in \mathcal{G}$. The conflict constraint (5.4) corresponds to (4.3). There is no constraint corresponding to (4.2) in IPM, as the columns are already defined as perfect matchings. On the other hand, Constraint (5.5) has no counterpart in MWPMC as it is a part of the redefinition of variables (5.2).

In each iteration of the B&P algorithm, the LP relaxation of IPM, the LP Master (LPM), is solved. When (5.6) is replaced with $\bar{\mathbf{0}}_{|\mathcal{G}|} \leq \boldsymbol{\lambda} \leq \bar{\mathbf{1}}_{|\mathcal{G}|}$, LPM is obtained. Due to Constraint (5.5), $\boldsymbol{\lambda} \leq \bar{\mathbf{1}}_{|\mathcal{G}|}$ term in the inequality is redundant and can be removed. As $|\mathcal{G}|$ is a very large number, it makes sense to solve LPM using column generation. Also, most of the variables in LPM is equal to zero in the optimal solution. We take advantage of this by solving the *Restricted LPM* (RLPM)

$$\text{RLPM:} \quad \max \quad \sum_{k=1}^{|\mathcal{T}|} (\mathbf{w}^T \bar{\mathbf{x}}_k) \lambda_k \tag{5.7}$$

$$\text{s.t.} \quad \sum_{k=1}^{|\mathcal{T}|} (\mathbf{B}_C^T \bar{\mathbf{x}}_k) \lambda_k \leq \bar{\mathbf{1}}_{|E(C)|} \quad : \boldsymbol{\pi} \tag{5.8}$$

$$\bar{\mathbf{1}}_{|\mathcal{T}|}^T \boldsymbol{\lambda} = 1 \quad : \mu \tag{5.9}$$

$$\boldsymbol{\lambda} \geq \bar{\mathbf{0}}_{\mathcal{T}}. \tag{5.10}$$

There are a smaller number of variables ($\mathcal{T}$) in RLPM. Hence, it can be solved to find an optimal solution to LPM. The dual variables $\boldsymbol{\pi}$ and μ are associated with the conflict constraints (5.8) and the convexity constraints (5.9), respectively.

Given dual variables $\boldsymbol{\pi}$ and μ, the reduced cost of a variable in RLPM is $(\mathbf{w}^T - \boldsymbol{\pi}^T\mathbf{B}_C^T)\bar{\mathbf{x}}_k - \mu$. As the RLPM is solved optimally, there exist no variables with positive reduced cost among the variables in the RLPM. However, such variables may exist in the variables of LPM that are not contained in the RLPM. Such variables may lead to a better solution. However, it is not possible to calculate the reduced costs of all variables in LPM a priori.

The variable with the largest reduced cost can be found by solving an optimization problem. Observe that the coefficient of the first term in the reduced cost, $\hat{\mathbf{w}}^T = \mathbf{w}^T - \boldsymbol{\pi}^T\mathbf{B}_C^T$, is multiplied with $\bar{\mathbf{x}}_k \in \mathcal{G}$. $\mathcal{G}$ was defined with respect to the perfect matching constraints. Hence, it is possible to find the maximum weighted element subject to the weights $\hat{\mathbf{w}}$ by solving the following subproblem

$$\begin{aligned} \text{SP:} \quad \max \quad & \hat{\mathbf{w}}^T\mathbf{x} && (5.11)\\ \text{s.t.} \quad & \mathbf{B}_G\mathbf{x} = \bar{\mathbf{1}}_{|V(G)|} && (5.12)\\ & \mathbf{x} \in \mathbb{B}^{|E(G)|}, && (5.13) \end{aligned}$$

which is an ordinary MWPMP.

When the optimal value of the SP is less than or equal to μ, there exists no item in $\mathcal{G}$ with a positive reduced cost. Hence, an optimal solution to RLPM is proven to be also optimal for LPM. Otherwise, a perfect matching with a positive reduced cost is found by solving SP. In this case, the optimal solution to SP is added as a new variable to RLPM. Let $\mathbf{x}^*$ be such an optimal solution for SP. Then, the variable $\lambda_{\mathcal{T}+1}$ that corresponds to $\mathbf{x}^*$ in RLPM is added with the new column of the coefficient matrix,

$$\begin{bmatrix} \mathbf{w}^T\mathbf{x}^* \\ \mathbf{B}_C^T\mathbf{x}^* \\ 1 \end{bmatrix}. \tag{5.14}$$

The main advantage of reformulating an IP as IPM, and solving LPM by column generation method is to find a tighter relaxation efficiently. An easy to solve SP formulation is required for efficiency as it takes many iterations to solve LPM. However, when the subproblem has *integrality property*, the solution of LPM is equal to the LP relaxation of the original IP [33]. Therefore, we cannot take advantage of the tighter relaxation in the proposed formulation, as the MWPMP possesses the integrality property.

This reformulation can be illustrated with the sample graph given in Figure 4.1 and its conflict graph given in Figure 2.6. There are two columns that can be generated from the sample graph, which are $\{1, 5\}$ and $\{3, 4\}$ with the associated incidence vectors $\bar{x}_1 = \{1, 0, 0, 0, 1\}$ and $\bar{x}_2 = \{0, 0, 1, 1, 0\}$, respectively.. Then, IPM becomes

$$\begin{aligned} \text{IPM:} \quad \max \quad & 14\lambda_1 + 15\lambda_2 && (5.15)\\ \text{s.t.} \quad & 2\lambda_1 + \lambda_2 \leq 1 && (5.16)\\ & \lambda_1 + \lambda_2 = 1 && (5.17)\\ & \lambda_1, \lambda_2 \in \{0, 1\}. && (5.18) \end{aligned}$$

The subproblem is an ordinary MWPMP, which does not contain constraint (4.26) with modified weights as proposed, and generates proposed columns.

5.1.2. The Stable-Set-Based Formulations

Formulating an IPM based on (4.3) by defining a generating set as $\mathcal{G} := \{\bar{\mathbf{x}} \in \mathbb{B}^{|E(G)|} : \mathbf{B}_C^T\bar{\mathbf{x}} \leq \bar{\mathbf{1}}_{|E(G)|}\}$ is also possible. Equations (5.2) govern the redefinition of the decision variables (4.4), where $\bar{\mathbf{x}}$ terms are the items of the new $\mathcal{G}$. The resulting model is

$$\text{RLPM:} \quad \max \quad \sum_{k=1}^{\mathcal{T}} (\mathbf{w}^T\bar{\mathbf{x}}_k)\lambda_k \tag{5.19}$$

$$\text{s.t.} \quad \sum_{k=1}^{\mathcal{T}} (\mathbf{B}_G\bar{\mathbf{x}}_k)\lambda_k = \bar{\mathbf{1}}_{|V(G)|} \quad : \boldsymbol{\pi} \tag{5.20}$$

$$\bar{\mathbf{1}}_{\mathcal{T}}^T\boldsymbol{\lambda} = 1 \quad : \mu \tag{5.21}$$

$$\boldsymbol{\lambda} \geq \bar{\mathbf{0}}_{\mathcal{T}}, \tag{5.22}$$

and contains only the matching constraints as $\mathcal{G}$ consist of the stable-sets of the conflict graph. In this formulation, the reduced costs of the variables are determined by the equation $(\mathbf{w}^T - \boldsymbol{\pi}^T\mathbf{B}_G)\mathbf{x} - \mu$, where $\hat{\mathbf{w}}^T = (\mathbf{w}^T - \boldsymbol{\pi}^T\mathbf{B}_G)$ are the weights of the edges in

$$\text{SP:} \quad \max \quad \hat{\mathbf{w}}^T\mathbf{x} \tag{5.23}$$

$$\text{s.t.} \quad \mathbf{B}_C^T\mathbf{x} \leq \bar{\mathbf{1}}_{|E(C)|} \tag{5.24}$$

$$\mathbf{x} \in \mathbb{B}^{|E(G)|}, \tag{5.25}$$

which is an MWSSP defined on the conflict graph C. As long as SP can produce columns with weights larger than μ, RLPM is reoptimized after adding the new column of the form,

$$\begin{bmatrix} \mathbf{w}^T\mathbf{x}^* \\ \mathbf{B}_G\mathbf{x}^* \\ 1 \end{bmatrix} \tag{5.26}$$

that corresponds $\boldsymbol{\lambda}_{\mathcal{T}+1}$ in the new constraint matrix.

A subproblem that does not have integrality property is MWSSP, which is $\mathcal{NP}$-hard. The advantage of this fact is that RLPM will produce tighter bounds than the LP relaxation of the original formulation. On the other hand, an $\mathcal{NP}$-hard problem has to be solved in each iteration for column generation.

We again consider the previous sample graph. A stable set in the conflict graph given in Figure 2.6 is $\{1, 2, 3, 4\}$ with the incidence vector $x_1 = \{1, 1, 1, 1, 0\}$. An RMLP that constains only x_1 is constructed as

$$\begin{aligned}
\text{RLPM:} \quad \max \quad & 26\lambda_1 && (5.27)\\
\text{s.t.} \quad & 3\lambda_1 = 1 && (5.28)\\
& 2\lambda_1 = 1 && (5.29)\\
& 2\lambda_1 = 1 && (5.30)\\
& \lambda_1 = 1 && (5.31)\\
& \lambda_1 = 1 && (5.32)\\
& \lambda_1 \in \{0, 1\}, && (5.33)
\end{aligned}$$

where (5.28)-(5.31) are the matching constraints and (5.32) is the convexity constraint.

It is also possible to develop a formulation based on the stable sets for the MWSSC with a similar approach. A generating set, which contains all stable-sets in $\hat{C}$, is defined with respect to (4.7) as $\mathcal{G} := \{\bar{\mathbf{x}} \in \mathbb{B}^{|E(G)|} : \mathbf{B}_C^T \bar{\mathbf{x}} \leq \bar{\mathbf{1}}_{|E(G)|}\}$. The resulting RLPM contains two constraints and it is formulated as

$$\begin{aligned}
\text{RLPM:} \quad \max \quad & \sum_{k=1}^{\mathcal{T}} (\mathbf{w}^T \bar{\mathbf{x}}_k)\lambda_k && && (5.34)\\
\text{s.t.} \quad & \sum_{k=1}^{\mathcal{T}} (\bar{\mathbf{1}}_{|E(G)|}^T \bar{\mathbf{x}}_k)\lambda_k = \frac{|V(G)|}{2} && : \pi && (5.35)\\
& \bar{\mathbf{1}}_{\mathcal{T}}^T \boldsymbol{\lambda} = 1 && : \mu && (5.36)\\
& \boldsymbol{\lambda} \geq \bar{\mathbf{0}}_{\mathcal{T}}. && && (5.37)
\end{aligned}$$

Constraint (5.35) ensures that the RLPM solution satisfies the cardinality restriction. The other constraint comes from Equation (5.2), which is a part of the reformulation of the variables.

The reduced costs are determined with $(\mathbf{w}^T - \pi\bar{\mathbf{1}}^T_{|E(G)|})\mathbf{x} - \mu$, where $\hat{\mathbf{w}}^T = (\mathbf{w}^T - \pi\bar{\mathbf{1}}^T_{|E(G)|})$ is the objective function coefficients in the following SP,

$$\text{SP:} \quad \max \quad \hat{\mathbf{w}}^T\mathbf{x} \tag{5.38}$$

$$\text{s.t.} \quad \mathbf{B}^T_{\hat{C}}\mathbf{x} \leq \bar{\mathbf{1}}_{|E(\hat{C})|} \tag{5.39}$$

$$\mathbf{x} \in \mathbb{B}^{|E(G)|}. \tag{5.40}$$

Similar to the previous formulations, the columns produced by SP are added to RLPM as their weights are larger than μ. Otherwise, the result of RLPM is also optimal for LPM. The column generated by SP is $\mathbf{x}^*$. Then, the new variable $\boldsymbol{\lambda}_{\mathcal{T}+1}$ is added with the newly generated coefficient column

$$\begin{bmatrix} \mathbf{w}^T\mathbf{x}^* \\ \bar{\mathbf{1}}^T_{|E(G)|}\mathbf{x}^* \\ 1 \end{bmatrix}, \tag{5.41}$$

whose weight with respect to $\mathbf{w}$ is the objective function coefficient, the number of edges in it is the cardinality constraint coefficient.

This formulation requires solving an MWSSP instance on $\hat{C}$ in each column generation iteration as a subproblem. As we discussed earlier, MWSSP on $\hat{C}$ is equivalent to solve an MWMC on G. These problems are also shown to be $\mathcal{NP}$-hard; nevertheless, they generate very tight results as only the cardinality restriction of the MWSSC is relaxed. Notice that, the columns in this formulation are stable-sets of the extended conflict graph.

For the extended conflict graph given in Figure 2.8, a subset of the stable-sets are $\{1\}$, $\{2\}$, and $\{3,4\}$. The columns corresponding to these elements are the variables of RLPM, which is given as

$$\begin{aligned} \text{RLPM:} \quad \max \quad & 5\lambda_1 + 6\lambda_2 + 15\lambda_3 && (5.42)\\ \text{s.t.} \quad & \lambda_1 + \lambda_2 + 2\lambda_3 = 2 && (5.43)\\ & \lambda_1 + \lambda_2 + \lambda_3 = 1 && (5.44)\\ & \lambda_1, \lambda_2, \lambda_3 \geq 0. && (5.45) \end{aligned}$$

5.2. The Partitioning-Based Schemes

The partitioning models of MWPMC (4.11)-(4.15) and MWSSC (4.16)-(4.20) has the constraints of the form shown in Figure 5.1. Observe that the conflict constraints corresponding to the vertex-induced subgraphs forms a block-diagonal structure. Hence, we propose P generating sets to redefine each decision variable vector in this section. This ends up with formulations with P subproblems which are handled separately.

$$\begin{bmatrix} \mathbf{B}_G(1) & \mathbf{B}_G(2) & \cdots & \mathbf{B}_G(P) \\ \mathbf{B}^T_{\tilde{C}}(1) & \mathbf{B}^T_{\tilde{C}}(2) & \cdots & \mathbf{B}^T_{\tilde{C}}(P) \\ \mathbf{B}^T_{C_1} & 0 & \cdots & 0 \\ 0 & \mathbf{B}^T_{C_2} & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & \mathbf{B}^T_{C_P} \end{bmatrix} \qquad \begin{bmatrix} \bar{\mathbf{1}}^T_{|V(\hat{C}_1)|} & \bar{\mathbf{1}}^T_{|V(\hat{C}_2)|} & \cdots & \bar{\mathbf{1}}^T_{|V(\hat{C}_P)|} \\ \mathbf{B}^T_{\hat{\tilde{C}}}(1) & \mathbf{B}^T_{\hat{\tilde{C}}}(2) & \cdots & \mathbf{B}^T_{\hat{\tilde{C}}}(P) \\ \mathbf{B}^T_{\hat{C}_1} & 0 & \cdots & 0 \\ 0 & \mathbf{B}^T_{\hat{C}_2} & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & \mathbf{B}^T_{\hat{C}_P} \end{bmatrix}$$

Figure 5.1. Constraint matrices of partitioning formulations.

For the partitioning based scheme of the MWPMC, the generating set containing all stable-sets in C_p is defined as $\mathcal{G}_p := \{\bar{\mathbf{x}} \in \mathbb{B}^{|V(C_p)|} : \mathbf{B}_{C_p}\bar{\mathbf{x}}_p \leq \bar{\mathbf{1}}_{|E(C_p)|}\}$. The decision variables $\bar{\mathbf{x}}_p$ (4.15) are redefined with the items in this set by the following equations.

$$\begin{aligned}
&\mathbf{x}_p = \sum_{k=1}^{|\mathcal{G}_p|} \bar{\mathbf{x}}_{pk}\lambda_{pk} \\
&\sum_{k=1}^{|\mathcal{G}_p|} \lambda_{pk} = 1 \\
&\lambda_{pk} = \{0,1\} \quad k = 1,2,\ldots,|\mathcal{G}_p|.
\end{aligned} \tag{5.46}$$

The following IPM is obtained by replacing the decision variable vectors with the corresponding redefinitions,

$$\text{IPM:} \quad \max \quad \sum_{p=1}^{P}\sum_{k=1}^{|\mathcal{G}_p|}(\mathbf{w}_p^T\bar{\mathbf{x}}_{pk})\lambda_{pk} \tag{5.47}$$

$$\text{s.t.} \quad \sum_{p=1}^{P}\sum_{k=1}^{|\mathcal{G}_p|}(\mathbf{B}_G(p)\bar{\mathbf{x}}_{pk})\lambda_{pk} = \bar{\mathbf{1}}_{|V(G)|} \tag{5.48}$$

$$\sum_{p=1}^{P}\sum_{k=1}^{|\mathcal{G}_p|}(\mathbf{B}_{\tilde{C}}^T(p)\bar{\mathbf{x}}_{pk})\lambda_{pk} \leq \bar{\mathbf{1}}_{|E(\tilde{C})|} \tag{5.49}$$

$$\bar{\mathbf{1}}_{|\mathcal{G}_p|}^T\boldsymbol{\lambda_p} = 1 \qquad p = 1,2,\ldots,P \tag{5.50}$$

$$\boldsymbol{\lambda_p} \in \mathbb{B}^{|\mathcal{G}_p|} \qquad p = 1,2,\ldots,P, \tag{5.51}$$

where (5.48) and (5.49) correspond to the matching constraints and the conflict constraints of $\tilde{C}$ in the original formulation. Both of them are not subject to the block-diagonal structure in Figure 5.1.

A similar method is used, which is replacing (5.51) with $\bar{\mathbf{0}}_{|\mathcal{G}_p|} \leq \boldsymbol{\lambda_p} \leq \bar{\mathbf{1}}_{|\mathcal{G}_p|}$, to obtain LPM. The inequality $\boldsymbol{\lambda_p} \leq \bar{\mathbf{1}}_{|\mathcal{G}_p|}$ is redundant again because of Equation (5.46). Solving LPM by column generation can be more efficient, as $|\mathcal{G}_p|$ can be a very large number. Hence, the following RLPM, which contains a smaller subset of variables of size $\mathcal{T}_p$, is solved in each column generation iteration.

$$\text{RLPM:} \quad \max \quad \sum_{p=1}^{P}\sum_{k=1}^{\mathcal{T}_p}(\mathbf{w}_p^T\bar{\mathbf{x}}_{pk})\lambda_{pk} \tag{5.52}$$

$$\text{s.t.} \quad \sum_{p=1}^{P}\sum_{k=1}^{\mathcal{T}_p}(\mathbf{B}_G(p)\bar{\mathbf{x}}_{pk})\lambda_{pk} = \bar{\mathbf{1}}_{|V(G)|} \quad : \boldsymbol{\pi} \tag{5.53}$$

$$\sum_{p=1}^{P}\sum_{k=1}^{\mathcal{T}_p}(\mathbf{B}_{\tilde{C}}^T(p)\bar{\mathbf{x}}_{pk})\lambda_{pk} \leq \bar{\mathbf{1}}_{|E(\tilde{C})|} \quad : \boldsymbol{\phi} \tag{5.54}$$

$$\bar{\mathbf{1}}_{\mathcal{T}_p}^T\boldsymbol{\lambda_p} = 1 \qquad p = 1, 2, \ldots, P \quad : \mu_p \tag{5.55}$$

$$\boldsymbol{\lambda_p} \geq \bar{\mathbf{0}}_{\mathcal{T}_p} \qquad p = 1, 2, \ldots, P. \tag{5.56}$$

An optimal solution to LPM can be found by optimizing RLPM. To check the optimality of an RLPM solution, no $\mathcal{G}_p$ must contain a column with a positive reduced cost. The reduced costs are calculated by $\hat{\mathbf{w}}_p^T\bar{\mathbf{x}}_{pk} - \mu_p$ for the decision variables corresponding to C_p, where $\hat{\mathbf{w}}_p^T = \mathbf{w}_p^T - \boldsymbol{\pi}^T\mathbf{B}_G(p) - \boldsymbol{\phi}^T\mathbf{B}_{\tilde{C}}^T$. Hence, we optimize P subproblems with weights $\hat{\mathbf{w}}_p$ subject to the conflict constraints defining $\mathcal{G}_p$. The p^{th} SP has the form,

$$\text{SP(p):} \quad \max \quad \hat{\mathbf{w}}_p^T\mathbf{x}_p \tag{5.57}$$

$$\text{s.t.} \quad \mathbf{B}_{C_p}\mathbf{x}_p \leq \bar{\mathbf{1}}_{|E(C_p)|} \tag{5.58}$$

$$\mathbf{x} \in \mathbb{B}^{|V(C_p)|}. \tag{5.59}$$

If the optimal solution of SP(p) is larger than μ_p, the optimal solution is a column with a positive reduced cost. Hence, it must be included in RLPM as a new column corresponding to the new variable $\boldsymbol{\lambda}_{p,\mathcal{T}+1}$. The form of the produced column is

$$\begin{bmatrix} \mathbf{w}_p^T \mathbf{x}_p^* \\ \mathbf{B}_G(p)\mathbf{x}_p^* \\ \mathbf{B}_{\bar{C}}^T(p)\mathbf{x}_p^* \\ 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix}. \tag{5.60}$$

Note that, only the coefficient in the p^{th} convexity constraint has value one, while the others are zero as we combine columns from each set separately. The optimal solution of RLPM is also optimal for LPM, when all the subproblems have nonpositive objective values.

The subproblems in this section are regular MWSSP solved over the vertex-induced subgraphs. Although it is an $\mathcal{NP}$-hard problem, solving it on the subgraphs is more efficient than solving it on the conflict graph, as proposed in the stable-set based formulation of MWPMC. On the other hand, this may increase the bound produced by LPM.

A possible partitioning of the conflict graph shown in Figure 2.6 is defined by vertex sets $V_1 = \{1, 2, 5\}$ and $V_2 = \{3, 4\}$. The stable-sets $\{1\}, \{2, 5\}$ and $\{3\}, \{3, 4\}$ can be generated from V_1 and V_2 respectively. These sets are contained as columns in RLPM, which is formulated as

$$\begin{aligned}
\text{RLPM:} \quad \max \quad & 5\lambda_{11} + 15\lambda_{12} + 7\lambda_{21} + 15\lambda_{22} && (5.61)\\
\text{s.t.} \quad & \lambda_{11} + \lambda_{12} + \lambda_{21} + \lambda_{22} = 1 && (5.62)\\
& \lambda_{11} + \lambda_{22} = 1 && (5.63)\\
& 2\lambda_{12} + \lambda_{22} = 1 && (5.64)\\
& \lambda_{12} + \lambda_{21} + \lambda_{22} = 1 && (5.65)\\
& \lambda_{11} + \lambda_{12} = 1 && (5.66)\\
& \lambda_{21} + \lambda_{22} = 1 && (5.67)\\
& \lambda_{11}, \lambda_{12}, \lambda_{21}, \lambda_{22} \geq 0. && (5.68)
\end{aligned}$$

In this example, (5.62)-(5.65) are the matching constraints. As the edge-induced subgraph in the proposed partitioning is an empty graph, there is no conflict constraints in RLPM. Finally, (5.66)-(5.67) are the convexity constraints for the columns generated from each partition. The subproblems are regular MWSSPs solved on the vertex-induced subgraphs.

For a partitioning reformulation of MWSSC, the generating sets are defined according to $\hat{C}_p$ as $\mathcal{G}_p := \{\bar{\mathbf{x}} \in \mathbb{B}^{|V(\hat{C}_p)|} : \mathbf{B}_{\hat{C}_p}\bar{\mathbf{x}}_p \leq \bar{\mathbf{1}}_{|E(\hat{C}_p)|}\}$. Equations (5.46) redefine the variables (4.20). The resulting RLPM is

$$\begin{aligned}
\text{RLPM:} \quad \max \quad & \sum_{p=1}^{P}\sum_{k=1}^{\mathcal{T}_p}(\mathbf{w}_p^T\bar{\mathbf{x}}_{pk})\lambda_{pk} && (5.69)\\
\text{s.t.} \quad & \sum_{p=1}^{P}\sum_{k=1}^{\mathcal{T}_p}(\bar{\mathbf{1}}_{|V(\hat{C}_p)|}^T\bar{\mathbf{x}}_{pk})\lambda_{pk} = \frac{|V(G)|}{2} \quad : \pi && (5.70)\\
& \sum_{p=1}^{P}\sum_{k=1}^{\mathcal{T}_p}(\mathbf{B}_{\tilde{\hat{C}}}^T(p)\bar{\mathbf{x}}_{pk})\lambda_{pk} \leq \bar{\mathbf{1}}_{|E(\tilde{\hat{C}})|} \quad : \boldsymbol{\phi} && (5.71)\\
& \bar{\mathbf{1}}_{\mathcal{T}_p}^T\boldsymbol{\lambda_p} = 1 \qquad p = 1, 2, \ldots, P \quad : \mu_p && (5.72)\\
& \boldsymbol{\lambda_p} \geq \bar{\mathbf{0}}_{\mathcal{T}_p} \qquad p = 1, 2, \ldots, P. && (5.73)
\end{aligned}$$

The reduced cost of the k^{th} element of set $\mathcal{G}_p$ is $\hat{\mathbf{w}}_p^T \bar{\mathbf{x}}_{pk} - \mu_p$ where $\hat{\mathbf{w}}_p = \mathbf{w}_p^T - \pi \bar{\mathbf{1}}^T_{|V(\hat{C}_p)|} - \boldsymbol{\phi}^T \mathbf{B}^T_{\tilde{\hat{C}}}$. The p^{th} subproblem maximizes $\hat{\mathbf{w}}_p$ coefficients over the $\mathcal{G}_p$ as follows

$$\begin{aligned} \text{SP(p):} \quad \max \quad & \hat{\mathbf{w}}_p^T \mathbf{x}_p && (5.74)\\ \text{s.t.} \quad & \mathbf{B}^T_{\hat{C}_p} \mathbf{x}_p \leq \bar{\mathbf{1}}_{|E(\hat{C}_p)|} && (5.75)\\ & \mathbf{x} \in \mathbb{B}^{|V(\hat{C}_p)|}. && (5.76) \end{aligned}$$

Note that, the subproblems are MWSSP on the vertex-induced subgraphs of the extended conflict graph. This is equivalent to solving MWMC on the edge-induced subgraph of the original graph, where the inducing edges correspond to the inducing vertices of the extended conflict graph.

A sample partitioning is proposed in Figures 4.2-4.4 for the extended conflict graph given in Figure 2.8. Here, possible stable-sets for the Partition 1 are $\{1\}, \{3\}$ while $\{4\}$ can be generated from Partition 2. These stable-sets are the columns of the following RLPM.

$$\begin{aligned} \text{RLPM:} \quad \max \quad & 5\lambda_{11} + 7\lambda_{12} + 8\lambda_{21} && (5.77)\\ \text{s.t.} \quad & \lambda_{11} + \lambda_{12} + \lambda_{21} = 2 && (5.78)\\ & \lambda_{11} + \lambda_{21} \leq 1 && (5.79)\\ & \lambda_{11} \leq 1 && (5.80)\\ & \lambda_{21} \leq 1 && (5.81)\\ & 0 \leq 1 && (5.82)\\ & \lambda_{12} \leq 1 && (5.83)\\ & \lambda_{11} + \lambda_{12} = 1 && (5.84)\\ & \lambda_{21} = 1 && (5.85)\\ & \lambda_{11}, \lambda_{12}, \lambda_{21} \geq 0. && (5.86) \end{aligned}$$

6. BRANCH-AND-PRICE ALGORITHMS

In general a B&P algorithm can be considered as a modification of the B&B framework, where the CG method is used to solve the LP relaxation in each node of the B&B tree. In this section, we first discuss the generic B&B framework. Then, we proposed B&P algorithms for solving MWPMC, which are based on the suggested Dantzig-Wolfe reformulations. Finally, primal heuristics, preprocessing methods, and an addition B&B-based MWMC algorithm that is used as a subproblem solver are proposed.

6.1. Branch-and-Bound

To solve IP problems optimally, B&B algorithm introduced in [34] can be utilized. The algorithm implicitly searches the solution space by bounding the solutions and branching on the space, as the name suggests. In this section, we first discuss the general outline of a B&B algorithm in the maximization setting.

Table 6.1 and Table 6.2 summarize the notation and the functions used in Algorithm 6.1. The algorithm is initialized with the IP to be solved, which is called the root node. The lower bound ($\underline{z}$) is the objective value of the incumbent, which is the best feasible solution found at a point in the algorithm. The lower bound and the incumbent are initialized to $-\infty$ and $\varnothing$ respectively since initially no feasible solution is known.

An active node from the search tree is selected at each iteration. Then, the relaxed solution is found to produce an upper bound. If the upper bound of the relaxation is less than the lower bound, the problem at hand cannot produce an integer solution better than the incumbent. Hence, the node is pruned by bound.

The solution of the relaxation may produce an integral solution. In this case, if the objective value is larger than the lower bound, the lower bound and the incumbent are updated. Otherwise, no further operations are done.

Finally, the upper bound may be larger than the lower bound. In this case, new nodes from the current node are produced by adding constraints that partition the solution space. This operation is called branching. The children nodes are added to the search tree. The whole process repeats until no more nodes to discover are left in the search-tree.

Table 6.1. Notation used in Algorithm 6.1.

Notation	**Definition**
R	Root Node, the IP to be solved
T	Search Tree, A list of nodes which are not yet discovered
N	Active node to be processed in the current iteration
S_N	The relaxed solution of N which produces the upper bound
$\bar{\mathbf{z}}_N$	Upper Bound of N, the objective value of S_N
Incumbent	The best known feasible solution to IP
$\underline{z}$	Lower Bound, the objective value of the incumbent

Table 6.2. Functions used in Algorithm 6.1.

Function	**Definition**
SelectNode(T)	Selects and deletes a node from the tree, returns the node
Relax(N)	Solves the relaxation to the given node, returns the solution
Check(S)	Checks whether a relaxed solution is a feasible solution. If the solution is integral, returns true.
Branch(N, S)	Creates child nodes from N with respect to solution S. Then, adds the child nodes to tree T.

Algorithm 6.1. Generic Branch-and-Bound Algorithm

Input: An IP Problem

Output: Optimal Objective Value and Solution

```
/* INITIALIZATION */
T = {(R = IP)}, z = −∞, Incumbent = ∅
/* TREE SEARCH */
while T ≠ ∅ do
  N = SelectNode(T)
  S_N, z̄_N = Relax(N)
  /* PRUNING */
  if z̄_N ≤ z then
    Continue
  end if
  /* LOWER BOUND UPDATE */
  if Check(S_N) then
    z = z̄_N
    Incumbent = S_N
    Continue
  end if
  /* BRANCHING */
  Branch(T, N, S_N)
end while
Return z and Incumbent
```

Figure 6.1. Generic branch-and-bound algorithm.

6.2. Branch-and-Price

The search in the B&B algorithm is mainly based on the comparison between the upper bound of a node and the lower bound produced by the best-known integer-valued solution. The gap between these two bounds has an impact on the number of nodes that must be solved. A tighter gap means fewer nodes to traverse.

Relaxation LPM may be tighter than the LP relaxation of the original IP. In fact, they are equal at the worst-case scenario. Hence, while solving IPM with B&P algorithm, it is expected to traverse fewer nodes than B&B algorithm solving the original IP.

We discuss the main differences of B&P, namely the column generation method, the branching rules, and the additional methods to increase the efficiency of the algorithm in the remaining subsections.

6.2.1. Column Generation

Algorithm 6.2 shows the steps of CG, which is used to solve the LPM relaxation of each node in B&P. Finding an initial feasible solution to LPM is the first step in CG. Then, the RLPM is solved and its optimality is controlled by solving subproblems. If there are columns with positive reduced costs, they are added to the RLPM. Until no more columns with positive reduced costs are left, this loop continues.

At each iteration, RLPM's optimum value $\{z^*_{\text{RLPM}}\}$ produces a lower bound on the optimum value $\{z^*_{\text{LPM}}\}$ of LPM. Also, an upper bound on LPM's optimum value can be calculated using subproblems' optimum solutions $\{z^*_{\text{SP(p)}},\ p = 1, 2, \ldots, P\}$ by

$$z^*_{\text{RLPM}} \leq z^*_{\text{LPM}} \leq z^*_{\text{RLPM}} + \sum_{p=1}^{P} (z^*_{\text{SP(p)}} - \mu_p) \tag{6.1}$$

When the gap between these bounds is closed, CG algorithm terminates. This is the point where no columns with a positive reduced cost can be generated.

6.2.1.1. Initial Feasible Solution. An initial feasible solution is required to start CG procedure. A problem-specific heuristic may produce such a solution. Alternatively, an artificial column that is assumed to comply with all constraints in RLPM could be used by the big-M or 2-phase methods.

Algorithm 6.2. Column Generation

Input: RLPM, SPs

Output: Optimal Objective Value and Solution for LPM

```
Find initial feasible solution, add it to RLPM
while TRUE do
    Solve(RLPM)
    Solve(SPs)
    if A candidate column is found then
        Introduce the new column to RLPM
    else
        Break
    end if
end while
Return RLPM Result and Solution
```

Figure 6.2. Column generation.

In the big-M method, a large cost is assigned to the artificial column in the objective function. If the problem is feasible, we expect the optimal solution not to contain the artificial column. Otherwise, we conclude that the problem is infeasible. The selection of big-M's value is crucial as it may lead to incorrect conclusions. An example of such problem is illustrated in [33].

The 2-phase approach overcomes the problem of selecting a correct big-M value by firstly eliminating the artificial variables in phase I. If there exist artificial variables at the end of phase I, the problem is infeasible. Hence, it can be pruned by infeasibility. If the artificial variables are eliminated at the end of the first phase, an initial feasible solution is found. One can completely remove the artificial variables from the problem in phase II. Alternatively, the artificial variables can be kept in the column pool with big-M costs in the second phase. This choice may increase the stability of the column generation algorithm [35].

The structure of the artificial variables is another choice for the initialization. A single artificial variable that represents a feasible solution to the whole RLPM may be used. In the multiple subproblem case, there may be an artificial variable for each subproblem, which represents feasible solutions to those problems. Finally, one may use an artificial variable for each RLPM constraint. A discussion on this topic is given in [35].

6.2.1.2. Acceleration Strategies. The optimality gap described by Equation (6.1) is quickly reduced in the initial iterations of CG applications. However, the convergence speed becomes considerable slower as the algorithm proceeds. Hence, a lot of iterations are required to close the gap. This phenomenon is called the tailing-off effect [35], which reduces the performance of the CG algorithm dramatically. The procedure can be terminated before the gap is completely closed to overcome this issue and the upper bound to the optimal value of LPM can be used as an upper bound to the optimal value of IPM.

The column with the most positive reduced cost is required to prove optimality and produce the upper bound. However, we do not have to add the column with the most positive reduced cost in each iteration. In fact, any column with the positive reduced cost proves that RLPM is not optimal and may be added to RLPM. Hence, if an efficient heuristic available to solve SPs, it can be used to speed up the procedure. Additionally, adding more than one column with positive reduced costs may also increase the performance.

There must exist no subproblems generating a column with positive reduced cost to prove optimality, if CG solves a formulation containing multiple subproblems. One may solve each subproblem in each column generation iteration until no more columns can be generated to satisfy this requirement. However, this may not be efficient. An alternative approach could be to solve some of the subproblems in each iteration. This technique is called partial pricing [33] and may accelerate the procedure.

6.2.2. Branching

Branching in B&B is done by adding constraints that partition the solution space into subspaces so that the infeasible solution produced by the relaxation is discarded. However, this approach may apply to an IPM. For instance, let λ_k be fixed to zero. Then, it will not be in any optimal solution of an RLPM. However, if the corresponding item $\bar{\mathbf{x}}_k$ has a positive reduced cost in a CG operation, it may appear as a new column represented by λ_{k+t}. On the other hand, when λ_k is fixed to one, the optimal solution would be directly determined in the formulations with a single subproblem due to the convexity constraints of the form (5.9).

An alternative approach for branching in B&P is branching on the original variables. After determining LPM's solution by CG, it can be transformed to the original variables' space using the redefinitions (5.2). If the solution is not integral in the original variable space, branching can be done on the original variables. The branching may be reflected in the B&P formulations in two ways, by RLPMs or by SPs.

Let $\mathbf{x}_i$ be fixed to a value $\mathbf{d}_i$ in the original space. If this operation is applied in the RLPM, firstly a constraint of the form $\sum_{k=1}^{\mathcal{T}}(\bar{\mathbf{x}}_{ik}\lambda_k) = \mathbf{d}_i$ must be added in the child node. The columns having $\bar{\mathbf{x}}_i \neq \mathbf{d}_i$ must be removed. Finally, the reduced cost formula must be updated with the dual value corresponding to the new constraint. After applying these operations, a solution conflicting with this constraint cannot be produced by RLPM. Even if a column with a positive reduced cost which has $\bar{\mathbf{x}}_i \neq \mathbf{d}_i$ is produced in SP, it cannot enter a feasible RLPM solution due to the new constraint.

Alternatively, the branching can be done via SPs. In this approach, there will not be any changes in RLPM and the reduced cost formulations, except removing the columns with $\bar{\mathbf{x}}_i \neq \mathbf{d}_i$. A new constraint of the form $\mathbf{x}_i = \mathbf{d}_i$ must be added to SP, as the original variables correspond to the SP variables. Hence, no new columns conflicting with the branching can be produced by SP. The branching partitions the solution space, as RLPM does not contain such columns as well.

Both approaches have advantages and disadvantages. For instance, when branching is done on RLPM, SP formulations do not change, which is an advantage when SP has a specific form that is efficiently solved. However, as the depth of the search tree increases, RLPM formulation becomes more complicated. On the other hand, the RLPM formulation does not change when the branching is done by SP, nevertheless, it may corrupt its form.

We branch on the original variables, then transform the branching constraints to the subproblems in all algorithms. The proposed subproblems are either MWPMP or MWSSP, which are solved on G, C, $\hat{C}$ or their partitions. Here the decision variables correspond to the edges or vertices of the corresponding graphs. The branching constraints require fixing the values of the edges or vertices. Hence, only the form of the graphs, on which the subproblems are solved, changes. Also, the columns that do not conform to the branching decisions are removed from RLPMs. As a result, both RLPM and subproblem formulations do not change when we apply this approach. Therefore, this rule is not affected by the previously mentioned drawbacks.

Another common branching method used in the context of B&P is the Ryan-Foster branching rule [28]. The rule is suitable for the problems whose IPM contains multiple subsystems and set partitioning constraints. When the subsystems in the IPM are identical, the rule is particularly effective due to the symmetry in such systems. In the case of distinct subsystems, such as the partitioning-based MWPMC formulation, the Ryan-Foster rule can be interpreted as the original variable branching as described in [4].

6.3. Preprocessing

A node is preprocessed before taken into consideration by Algorithm 6.3. The preprocessing may prune a node, may decrease the size of a subproblem, or may not effect. Three tests are applied to a node. The first test checks whether there exists a sufficient number of edges in a node.

A node is defined according to three sets of edges in a graph, which are included, excluded, and free edges. The included set is the set of edges that are fixed to be in a perfect matching. The excluded set on the other hand is fixed to be out of a perfect matching. Finally, the remaining edges, namely the ones that are neither included nor excluded form the subset of free edges, and we decide what to do with them.

Clearly, $\frac{|V(G)|}{2}$ independent edges create a perfect matching. Therefore, we come up with the condition

$$|E(G)| - |I| - |X| = |F| \geq \frac{|V(G)|}{2} - |I| \tag{6.2}$$

for feasibility. This is clearly equivalent to

$$|X| \leq |E(G)| - \frac{|V(G)|}{2}. \tag{6.3}$$

Hence, we need at least $\frac{|V(G)|}{2} - |I|$ free edges of a search node. If the number of free edges is smaller, then there exist no perfect matchings in the current node. Hence, the node can be pruned by infeasibility. This procedure is called the edge number test.

If the node passes the edge number test, we apply a fast upper bound test, which is computationally cheap. An upper bound on a node can be found by the sum of the weights of the largest $\frac{|V(G)|}{2} - |I|$ added to the weights in the included set. If this upper bound is less than the lower bound, the node can be pruned.

Finally, we apply the third test, which is called the logical test. This test firstly searches the vertex with the smallest degree. If it is equal to zero, the node is isolated, which means it is not possible to produce a perfect matching. Hence, the node is pruned. If the smallest degree is equal to one, the edge incident to the vertex should be in the perfect matching. Hence, the edge is added to the included set, and the edges that conflict with that edge are excluded. Then, the procedure restarts and continues until no more changes occur.

Algorithm 6.3. Preprocess

Input: A Node

Output: The preprocessed node

```
Apply Edge Number Test
Apply Fast Upper Bound Test
/* Logical Test */
while true do
  v = SelectMinDegreeVertex()
  if d_G(v) = 0 then
    return false
  else if d_G(v) = 1 then
    e = δ(v)
    include(e)
    continue
  else
    break
  end if
end while
if there are changes in the logical test then
  Apply Edge Number Test
  Apply Fast Upper Bound Test
end if
return true
```

Figure 6.3. Preprocess.

6.4. Primal Heuristics

We propose a greedy stable-set heuristic, given as Algorithm 6.4, on the extended conflict graph $\hat{C}$ to find a conflict-free matching on the original graph G. Here, the vertices of $\hat{C}$ are partitioned into free, included, and excluded sets, which correspond to the edge sets of G that define a node.

The algorithm firstly assign priority values to the free vertices of $\hat{C}$. In each iteration, it selects the one with the smallest value and adds it to the included set. The neighbors of this vertex are added to the excluded set and both the included vertex and its neighbors in $\hat{C}$ are removed from the free set. The algorithm terminates when there are no more elements left in the free set. At the end, the included set corresponds to a conflict-free matching in the original graph. If the matching is perfect and has a weight larger than the lower bound, the incumbent can be updated by this perfect matching.

The prioritization of the vertices may be done concerning several criteria. If the vertex degrees are used for prioritization, the heuristic works towards finding a matching of larger size. However, as this approach does not consider the weights, the resulting matching may have a small weight. On the contrary, the weights may be used as priorities, which tends to increase the total weight of the matching. Nevertheless, being agnostic to the vertex degrees reduces the chance of finding a perfect matching. A combination of these two criteria, such as the decreasing order of the weight-degree ratios, may be used to emphasize finding a perfect matching with a good weight.

Algorithm 6.4. Greedy Stable Set Heuristic

Input: Extended Conflict Graph ($\hat{C}$)

Output: A Stable Set on the Extended Conflict Graph

Grade all vertices

$F = V(\hat{C})$, $I = \varnothing$, $X = \varnothing$

while $F \neq \varnothing$ **do**

v = *SelectLeastPriorVertex*()

$I = I \cup v$

$X = X \cup \delta(v)$

$F = F/\delta[v]$

end while

return I

Figure 6.4. Greedy stable-set heuristic.

6.5. An Alternative Subproblem Solver

Solving MWSSP on $\hat{C}$, which is equivalent to solving MWMC on G, or on the partitions of $\hat{C}$ is required for MWSSC-based formulations. CPLEX shows a poor performance on both MWSSP and MWMC, which reduces the efficiency of the overall B&P algorithm. Hence, we propose an efficient B&B algorithm for MWMC, which is built on Algorithm 6.1.

The emphasis of this algorithm is on finding a conflict-free matching of high cost as it is used as a subproblem solver. As soon as it finds a column with a positive reduced cost, the optimization process is terminated to introduce the column to RLPM. Alternatively, one may search for multiple such columns. The depth-first search method for tree search is used for this reason. *SelectNode*() function in the Algorithm 6.1 is designed accordingly. After selecting a node to solve, the Algorithm 6.4 with an emphasis on the weight is run to increase the lower bound if possible.

In *Relax*() method, the MWM solver in LEMON graph and network optimization library [36] is run to find a relaxed solution. If the node is not pruned by bound, *Check*() function controls whether the MWM solution contains any conflicting pair. If the solution is conflict-free, the incumbent and lower bound are updated. Otherwise, the edge with the largest number of conflicts is selected as the branching variable. This criterion is determined to eliminate the largest number of conflicts in the child nodes.

7. COMPUTATIONAL EXPERIMENTS AND RESULTS

The proposed algorithms are implemented and tested on a large test problem set. The results are compared with each other and with benchmarks, which are obtained from formulations that are run on a state-of-the-art commercial solver. In this chapter, we discuss the details of the implementation, the tests, and their results.

7.1. Implementation

We implemented B&P algorithms based on all five Dantzig-Wolfe reformulations proposed in Chapter 5. These are the perfect-matching-based MWPMC, the stable-set-based MWPMC and MWSSC, and finally the partitioning-based MWPMC and MWSSC reformulations. All the programs are coded from scratch by following B&B framework given in Algorithm 6.1. The notation given in Table 6.1 and functions given in Table 6.2 except *Relax*(N) are common in all implementations. Moreover, the preprocess function given in Algorithm 6.3 and the stable-set heuristic given in Algorithm 6.4 does not depend on the formulation. Hence, the same codes are used for the implementation of these structures and functions.

The only difference among the programs is the *Relax*(N) function, the structure of which is given in Algorithm 6.2 for B&P algorithms. Each program requires solving an RLPM in each CG iteration. We use CPLEX LP solver [37] to optimize RLPMs, which are formulated with respect to the Dantzig-Wolfe reformulation at hand. As a result, the difference between the programs boils down to formulations of RLPMs, structures of the subproblems, and algorithms that are utilized to solve them.

The perfect-matching-based MWPMC, the stable-set-based MWPMC and the stable-set-based MWSSC reformulations require solving a single subproblem. The subproblem in the perfect-matching-based MWPMC is an ordinary MWPMP. We use the MWPMP solver available in LEMON graph and network optimization library [36].

In the stable-set-based MWPMC, an MWSSP on a conflict graph C must be solved as a subproblem. We assume that conflict graph C is an arbitrary graph. Therefore, BIP formulation of MWSSP is solved by CPLEX MIP solver in each iteration. The solver has difficulty in optimizing large instances as MWSSP is $\mathcal{NP}$-hard. On the other hand, MWSSP is polynomially solvable on some graph classes such as perfect graphs [38] and line graphs [2]. If the graph class, which C belongs to, is among these classes, a polynomial time algorithm can be utilized to solve the subproblems.

The stable-set-based MWSSC formulation also requires solving an MWSSP instance on the extended conflict graph $\hat{C}$ in each CG iteration. As CPLEX is inefficient while solving MWSSP, we propose an alternative B&B algorithm in Chapter 6. The framework given in Algorithm 6.1 is followed to code B&B algorithm. For the *Relax*(N) function, we utilized MWM solver available in LEMON. The rest of the code utilizes the common structures and functions provided in Table 6.1 and Table 6.2.

The partitioning-based functions require obtaining partitions from C and $\hat{C}$. The objective in graph partitioning is to minimize the number of edges in the edge-induced subgraph while keeping the sizes of the vertex-induced subgraphs as close as possible. The graph partitioning problem is $\mathcal{NP}$-complete [39]. However, there exist efficient heuristics for graph partitioning and open-source codes utilizing them such as METIS [40]. We utilized METIS for the partitioning of the graphs in the partitioning-based formulations. Only a single partitioning is done before beginning and the same partitions are kept using throughout B&P iterations. The subproblems are solved by CPLEX in both formulations due to the ease of implementation via the solver's API.

Finally, all algorithms are implemented in C++ [41]. We also coded programs to run BIP formulations of MWPMC and MWSSC on CPLEX v12.10 using CPLEX C++ API. We run CPLEX in both single-thread and multi-thread modes where dynamic search [42] is turned off and optimality is emphasized [43]. For all instances, the default time limit of 1800 seconds is used. The tests are run on a machine configured with an Intel Core i7-1065G7 processor, 16 GB RAM and running on Windows 10.

7.2. Tests & Results

The test set contains instances with $|V(G)| \in \{20, 40, \ldots, 100\}$. For each of these sets, there exist instances with graph densities $\Gamma \in \{0.2, 0.5, 0.8\}$, where $\Gamma = \frac{|E(G)|\times 2}{|V(G)|\times(|V(G)-1|)}$. Moreover, for each of these sets, the conflict graphs with densities $\Omega \in \{0.2, 0.4, 0.6, 0.8\}$, where $\Omega = \frac{|E(C)|\times 2}{|E(G)|\times(|E(G)-1|)}$, are contained in the set. Finally, there exist 5 instances for each combination of these settings in the test set, which contains 300 instances in total.

7.2.1. CPLEX Results

First of all, strong and weak formulations of MWPMC and MWSSC are solved by CPLEX in single-thread mode to obtain a base performance scenario for each instance in the test set. Moreover, the best performing formulations are also run in multi-thread mode. Table 7.1 shows a summary of these runs. The first column indicates the problem type, the formulation type, and the solver mode. The rest of the columns shows the number of instances for each stopping situation. Namely, the second column shows the number of instances proven optimal within the time limit of 1800 seconds for each setting. The numbers of instances for which at least an integer solution is found but not proven optimal are shown in the third column. Finally, the fourth one displays the number of instances that are not proven infeasible, but for which a feasible solution is also not found.

We refer to the column names, which are optimal, feasible, and unknown, to define the stopping situations and to refer to the instances in these groups in the remainder. For example, when we say that an algorithm returns optimal or an instance is optimal, then the algorithm completes the search and finds an optimal solution for the particular instance. Similarly, returning feasible or being feasible refer to the situation that a feasible solution, which is not proven optimal, is found within the time limit. If no feasible solution is found within the time limits, then we say that the algorithm returns unknown while the instance is unknown.

Table 7.1. CPLEX results summary.

Setting	Optimal	Feasible	Unknown	Total
MWPMCStrongSingle	205	88	7	300
MWPMCWeakSingle	210	88	2	300
MWSSCStrongSingle	141	129	30	300
MWSSCWeakSingle	147	115	38	300
MWPMCStrongMulti	198	90	12	300
MWPMCWeakMulti	210	88	2	300

It can be observed that MWPMC weak formulation shows the best performance in both single and multi-threaded runs in terms of the number of instances that are solved optimally. A counter-intuitive result is that MWPMC strong formulation has a more promising performance in single-threaded mode than the multi-threaded mode. This may be due to the overhead in parallelization of the underlying Branch & Cut algorithm. The performance of MWSSC formulations is not promising in single-threaded runs. Hence, we do not include multi-threaded results for them. The single-threaded results for MWPMC strong formulation are similar to the results of MWPMC weak formulation with respect to the number of optimal and feasible results. Among the instances that are proven optimal, 135 instances are solved optimally by all settings, while this number increases to 197 when we exclude MWSSC. The number of instances that are returned a feasible solution by all settings is 55 in total, and 74 only for MWPMC settings. Finally, none of the unknown instances are common for all settings.

Figure 7.1 contains the performance profiles, which is the method described in [44], of all CPLEX settings. At first, the best performing setting is determined for all instances in the test set in terms of runtimes. Then, the ratio of each setting's runtime to the best setting's runtime is calculated. The figure displays the fraction of instances that are in the following range

$$\text{the best runtime} \leq \text{the setting's runtime} \leq 2^x \times \text{the best runtime}. \tag{7.1}$$

The values in the x-axis of Figure 7.1 correspond to x values in Equation (7.1). For example, the figure shows the fraction of instances where an algorithm performs best, when x equals to 0. Hence, while MWPMCWeakMulti setting, which is shown by purple dashed line, solves around 27% of instances most efficiently, this value is less than 5% for MWSSCStrongSingle, MWSSCWeakSingle and MWPMCStrongMulti settings, which are shown by orange solid, blue dashed and black dashed lines respectively.

When x in Equation (7.1) is equal to two, MWPMCWeakSingle and MWPMCWeakMulti settings increase to approximately 70%. In other words, these two setting are in the range of [the best runtime, $4 \times$ the best runtime] in almost all instances that they are solved optimally. Also, the figure displays that the best performing formulation is MWPMCWeak formulation in both settings. It confirms that the MWPMCSingle formulation performs better in single-thread mode. Lastly, MWSSC formulations shows the worst performance among the formulations in Figure 7.1.

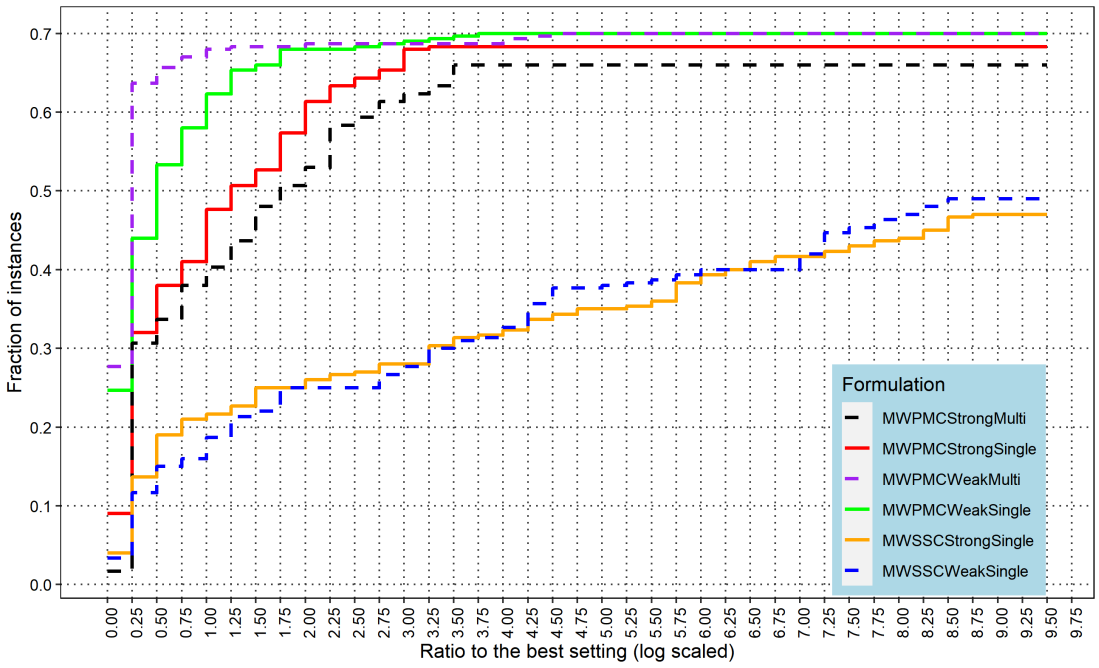

Figure 7.1. Performance profiles of CPLEX formulations.

Table 7.2 and 7.3 contains the average solution times, where the complete set of results are given in Appendix A.1. The rows represent the algorithm results for each set of vertex numbers indicated in the left-most column. The columns correspond to the density of the graph and the conflict graph in Table 7.2 and Table 7.3, respectively. All formulations solve the small instances that have 20 vertices effectively. However, the results coming from the instances having at least 40 vertices clearly show that MWSSC-based formulations spend significantly more time than MWPMC-based formulations. While the best performing setting changes with respect to the instance configuration, MWPMC-based formulations show similar performance to each other on the average. Table 7.2 displays that average time spent increases as the graph density increases. On the other hand, the conflict graph density has an opposite effect on the average time spent according to Table 7.3.

7.2.2. Branch-and-Price Results

The implemented algorithms are firstly compared to each other in a small subset of the test set which contains only the instances with $|V(G)| \in \{20, 40, 60\}$ vertices. This pre-assessment allows us to continue with the algorithms that perform better. Table 7.4 contains the number of instances that are solved optimally in the Optimal column. The numbers in the Feasible and Unknown columns show the number of instances that are not proven optimal in the given CPU time limit. While at least an integer solution is found for the instances in the feasible column, the algorithms cannot find any solution to the unknown instances. The table also contains CPLEX results in the subset for comparison.

Table 7.2. The effect of graph density on the solver.

$\|V(G)\|$	Formulation	$\Gamma = 0.2$	$\Gamma = 0.5$	$\Gamma = 0.8$	Avg.
20	MWPMCStrongMulti	0.01	0.05	0.15	0.07
	MWPMCStrongSingle	0	0.09	0.51	0.2
	MWPMCWeakMulti	0.01	0.04	0.11	0.05
	MWPMCWeakSingle	0	0.06	0.29	0.12
	MWSSCStrongSingle	0	0.25	1.51	0.59
	MWSSCWeakSingle	0.01	0.43	1.33	0.59
	Avg.	0	0.15	0.65	0.27
40	MWPMCStrongMulti	0.06	54.77	550.11	201.65
	MWPMCStrongSingle	0.06	120.13	519.56	213.25
	MWPMCWeakMulti	0.02	38.86	488.5	175.79
	MWPMCWeakSingle	0.02	73.61	516.39	196.67
	MWSSCStrongSingle	0.62	575.86	1368.03	648.17
	MWSSCWeakSingle	6.35	776.01	1005.45	595.94
	Avg.	1.19	273.21	741.34	338.58
60	MWPMCStrongMulti	4.08	944.85	1355.32	768.08
	MWPMCStrongSingle	4.17	734.45	1186.29	641.64
	MWPMCWeakMulti	1.38	600.6	1112.06	571.35
	MWPMCWeakSingle	1.75	697.78	1165.42	621.65
	MWSSCStrongSingle	53.34	1432.13	1800.08	1095.18
	MWSSCWeakSingle	547.05	1374.41	1610.21	1177.22
	Avg.	101.96	964.04	1371.56	812.52
80	MWPMCStrongMulti	173.58	1364.03	1800.79	1112.8
	MWPMCStrongSingle	211.63	1359.66	1800.02	1123.77
	MWPMCWeakMulti	42.24	1138.92	1800.45	993.87
	MWPMCWeakSingle	84.07	1270.63	1800.03	1051.58
	MWSSCStrongSingle	1091.65	1800.09	1800.29	1564.01
	MWSSCWeakSingle	970.28	1789.47	1800.04	1519.93
	Avg.	428.91	1453.8	1800.27	1227.66
100	MWPMCStrongMulti	739.85	1378.05	1800.73	1306.21
	MWPMCStrongSingle	572.16	1369.29	1805.29	1248.91
	MWPMCWeakMulti	488.12	1708.02	1800.5	1332.21
	MWPMCWeakSingle	498.23	1494.8	1800.09	1264.37
	MWSSCStrongSingle	1441.47	1800.29	1800.68	1680.81
	MWSSCWeakSingle	1390.14	1800.05	1800.12	1663.44
	Avg.	854.99	1591.75	1801.23	1415.99

Table 7.3. The effect of conflict graph density on the solver.

$\|V(G)\|$	Formulation	$\Omega = 0.2$	$\Omega = 0.4$	$\Omega = 0.6$	$\Omega = 0.8$	Avg.
20	MWPMCStrongMulti	0.1	0.1	0.06	0.02	0.07
	MWPMCStrongSingle	0.29	0.45	0.06	0.02	0.2
	MWPMCWeakMulti	0.08	0.08	0.03	0.02	0.05
	MWPMCWeakSingle	0.18	0.25	0.02	0.01	0.12
	MWSSCStrongSingle	0.34	1.46	0.54	0.01	0.59
	MWSSCWeakSingle	0.8	1.09	0.45	0.01	0.59
	Avg.	0.3	0.57	0.19	0.01	0.27
40	MWPMCStrongMulti	667.03	118.68	18.93	1.94	201.65
	MWPMCStrongSingle	751.27	85.08	15.31	1.34	213.25
	MWPMCWeakMulti	649.22	38.8	13.74	1.41	175.79
	MWPMCWeakSingle	694.01	72.71	18.67	1.31	196.67
	MWSSCStrongSingle	1179.23	741.16	644.16	28.14	648.17
	MWSSCWeakSingle	1207.42	1023.2	142.87	10.26	595.94
	Avg.	858.03	346.6	142.28	7.4	338.58
60	MWPMCStrongMulti	1205.61	1147.95	708.68	10.1	768.08
	MWPMCStrongSingle	1204.96	915.67	438.37	7.54	641.64
	MWPMCWeakMulti	1202.81	768.77	312.05	1.76	571.35
	MWPMCWeakSingle	1202.12	887.4	392.94	4.14	621.65
	MWSSCStrongSingle	1242.53	1223.32	1202.33	712.55	1095.18
	MWSSCWeakSingle	1800.02	1320.41	1205.5	382.96	1177.22
	Avg.	1309.67	1043.92	709.98	186.51	812.52
80	MWPMCStrongMulti	1417.87	1211.08	1201.83	620.43	1112.8
	MWPMCStrongSingle	1464.69	1215.88	1200.34	614.18	1123.77
	MWPMCWeakMulti	1251.63	1205.61	916.43	601.83	993.87
	MWPMCWeakSingle	1300.48	1210.75	1093.39	601.69	1051.58
	MWSSCStrongSingle	1800.03	1800.18	1423.88	1231.96	1564.01
	MWSSCWeakSingle	1800.01	1800.03	1284.91	1194.78	1519.93
	Avg.	1505.78	1407.25	1186.8	810.81	1227.66
100	MWPMCStrongMulti	1800.21	1579.37	1203.95	641.29	1306.21
	MWPMCStrongSingle	1800.02	1359.25	1200.93	635.45	1248.91
	MWPMCWeakMulti	1800.85	1250.35	1200.95	1076.69	1332.21
	MWPMCWeakSingle	1800.03	1263.64	1200.63	793.19	1264.37
	MWSSCStrongSingle	1800.15	1800.29	1800.42	1322.38	1680.81
	MWSSCWeakSingle	1800.04	1800.06	1800.05	1253.61	1663.44
	Avg.	1800.22	1508.83	1401.16	953.77	1415.99

According to the table, stable-set-based and partitioning-based MWPMC algorithms show worse performance than other algorithms considering the number of optimal solutions found. Matching-based MWPMC algorithm performs relatively close to MWSSC formulations solved by CPLEX. Finally, the stable-set-based MWSSC algorithm, which utilizes B&B algorithm for MWMC for solving the subproblems, shows a relatively close performance to the best CPLEX formulations considering the number of optimal results.

The number of feasible instances is low, as opposed to the number of unknowns. This may be explained as when a good feasible solution is found, the algorithm tends to prove optimality rapidly. Finally, Partitioning-Based-MWPMC optimally solves 105 instances, which is the smallest number instances that are proven optimal. All of these instances are proven optimal by all the other algorithms. 16 instances are returned unknown by all algorithms, while one instance is returned feasible by all. The remaining 59 instances are returned differently by distinct algorithms.

Table 7.4. B&P results summary.

Algorithm	Optimal	Feasible	Unknown	Total
Matching-Based-MWPMC	126	29	25	180
Partitioning-Based-MWPMC	105	45	30	180
Partitioning-Based-MWSSC	115	36	29	180
Stable-Set-Based-MWPMC	105	44	31	180
Stable-Set-Based-MWSSC	157	5	18	180
MWPMCStrongMulti	153	25	2	180
MWPMCStrongSingle	160	20	0	180
MWPMCWeakMulti	160	19	1	180
MWPMCWeakSingle	160	18	2	180
MWSSCStrongSingle	126	40	14	180
MWSSCWeakSingle	130	30	20	180

Figure 7.2 displays the performance profiles of all B&P implementations and a subset of CPLEX formulations. All CPLEX results are not given for the sake of readability of the plot. The line named as BPSSMWSSC corresponds to the stable-set based MWSSC formulation, which shows the best performance among all B&P implementations. Moreover, it shows the best performance in more than 50% of the instances, including the CPLEX formulations' results. According to the figure, BPSSMWSSC initially shows a competitive performance with MWPMCWeak formulation run on CPLEX in multi-thread setting and better performance than MWPMCStrong formulation run on CPLEX in single-thread mode.

The stable-set based MWPMC implementation (BPSSMWPMC) and the partitioning based MWSSC implementation (BPPartMWSSC) show similar performance to MWSSC formulation run on CPLEX. Finally, the partitioning based MWPMC implementation (BPPartMWPMC) and the stable-set based MWPMC implementation (BPSSMWPMC) show similar performance to each other and worse performance than the other implementations, according to the figure.

Tables 7.5 and 7.6 show the CPU time averages for each algorithm with respect to the graph and conflict graph densities. The raw computational test results are contained in Appendix A.2. It can be observed that both Partitioning-Based MWPMC and MWSSC algorithms and Stable-Set-Based MWPMC algorithm have difficulty in solving even the smallest set of instances that have 20 vertices. Stable-Set-Based MWSSC and Matching-Based MWPMC algorithms can solve these instances efficiently. However, when the number of vertices increases, the performance of Matching-Based MWPMC algorithm decreases dramatically. For instance, the average CPU runtime over the instances with 40 vertices and 0.6 conflict density is 597.88 seconds for Matching-Based MWPMC algorithm while it is only 1.51 seconds for Stable-Set-Based MWSSC algorithm, according to Table 7.6. Hence, we can conclude that the best performing B&P algorithm is Stable-Set-Based MWSSC. Finally, increasing graph density negatively affects the performance, while the opposite is true for the conflict graph density.

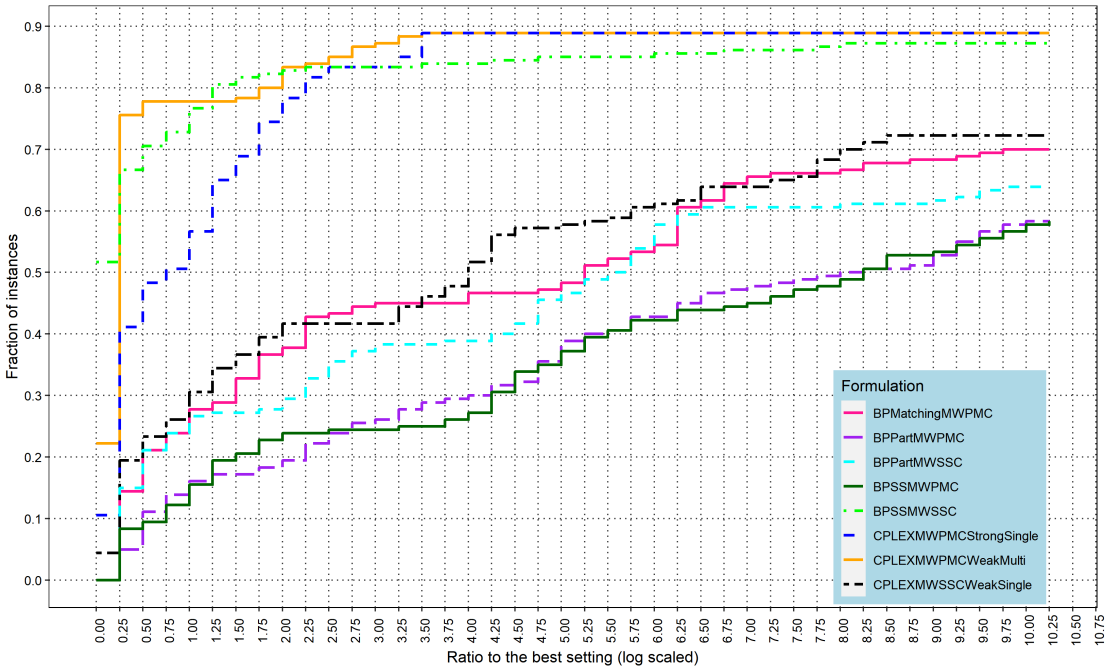

Figure 7.2. Performance profiles of B&P and CPLEX.

Table 7.5. The effect of graph density on B&P algorithms.

$\|V(G)\|$	Algorithm	$\Gamma = 0.2$	$\Gamma = 0.5$	$\Gamma = 0.8$	Avg.
20	Matching-Based-MWPMC	0.01	0.35	2.54	0.97
	Partitioning-Based-MWPMC	0.22	13.73	227.84	80.6
	Partitioning-Based-MWSSC	0.07	10.06	191.18	67.1
	Stable-Set-Based-MWPMC	0.17	15.13	102.13	39.14
	Stable-Set-Based-MWSSC	0	0.01	0.04	0.02
	Avg.	0.09	7.86	104.75	37.57
40	Matching-Based-MWPMC	1.95	652	1355.03	669.66
	Partitioning-Based-MWPMC	32.39	1159.25	1802.14	997.93
	Partitioning-Based-MWSSC	13.12	911.64	1452.97	792.58
	Stable-Set-Based-MWPMC	194.86	1218.2	1800.24	1071.1
	Stable-Set-Based-MWSSC	0.15	439.72	504.19	314.69
	Avg.	48.5	876.16	1382.91	769.19
60	Matching-Based-MWPMC	235.65	1608.74	1801.55	1215.31
	Partitioning-Based-MWPMC	572.88	1808.9	1816.15	1399.31
	Partitioning-Based-MWSSC	476.07	1378.25	1815.32	1223.21
	Stable-Set-Based-MWPMC	554.33	1800.21	1800.54	1385.03
	Stable-Set-Based-MWSSC	191.92	744.25	963.31	633.16
	Avg.	406.17	1468.07	1639.37	1171.2
Overall Avg.		151.59	784.03	1042.34	659.32

Table 7.6. The effect of conflict graph density on B&P algorithms.

$\|V(G)\|$	Algorithm	$\Omega = 0.2$	$\Omega = 0.4$	$\Omega = 0.6$	$\Omega = 0.8$	Avg.
20	Matching-Based-MWPMC	1.68	1.37	0.56	0.26	0.97
	Partitioning-Based-MWPMC	277.85	26.97	10.36	7.23	80.6
	Partitioning-Based-MWSSC	244.05	22.87	1.46	0.03	67.1
	Stable-Set-Based-MWPMC	110.46	31.12	7.6	7.4	39.14
	Stable-Set-Based-MWSSC	0.03	0.03	0.01	0	0.02
	Avg.	126.81	16.47	4	2.98	37.57
40	Matching-Based-MWPMC	1200.81	832.45	597.88	47.49	669.66
	Partitioning-Based-MWPMC	1240.19	1203.65	811.49	736.38	997.93
	Partitioning-Based-MWSSC	1216.99	1200.43	613.41	139.48	792.58
	Stable-Set-Based-MWPMC	1459.58	1201.93	971.8	651.09	1071.1
	Stable-Set-Based-MWSSC	1181.8	75.25	1.51	0.19	314.69
	Avg.	1259.87	902.74	599.22	314.93	769.19
60	Matching-Based-MWPMC	1461.25	1226.34	1215.1	958.55	1215.31
	Partitioning-Based-MWPMC	1805.21	1350.78	1223.21	1218.03	1399.31
	Partitioning-Based-MWSSC	1802.48	1226.76	1217.27	646.33	1223.21
	Stable-Set-Based-MWPMC	1802.63	1316.51	1214.57	1206.4	1385.03
	Stable-Set-Based-MWSSC	1455.41	986.27	86.84	4.11	633.16
	Avg.	1665.4	1221.33	991.4	806.68	1171.2
Overall Avg.		1017.36	713.52	531.54	374.86	659.32

7.2.3. The Comparison of the Best Performing Algorithms

The best performing formulations and algorithms, namely MWPMCStrongSingle, MWPMCWeakMulti formulations, and stable-set-based MWSSC algorithm, are tested on the complete test set. A summary is reported in Table 7.7. According to the table, the B&P implementation can solve more problems optimally than the formulations solved by CPLEX. However, the number of unknown instances is also much higher, which follows the earlier comment on this issue. 196 instances are solved optimally by all three methods.

Table 7.7. The comparison of CPLEX and B&P.

Algorithm	Optimal	Feasible	Unknown	Total
MWPMCStrongSingle	205	88	7	300
MWPMCWeakMulti	210	88	2	300
Stable-Set-Based-MWSSC	212	29	59	300

Figure 7.3 shows the performance profile of the best methods. Stable-set based MWSSC implementation shows the best performance in approximately 35% of all instances, while this value is around 25% and 15% respectively for MWPMCWeak and MWPMCStrong formulations run on CPLEX. Moreover, B&P implementation shows a competitive performance to MWPMCWeak formulation and a better performance than MWPMCStrong formulation in the rest of the figure. Hence, we can conclude that the implemented algorithm is competitive to the best CPLEX formulations.

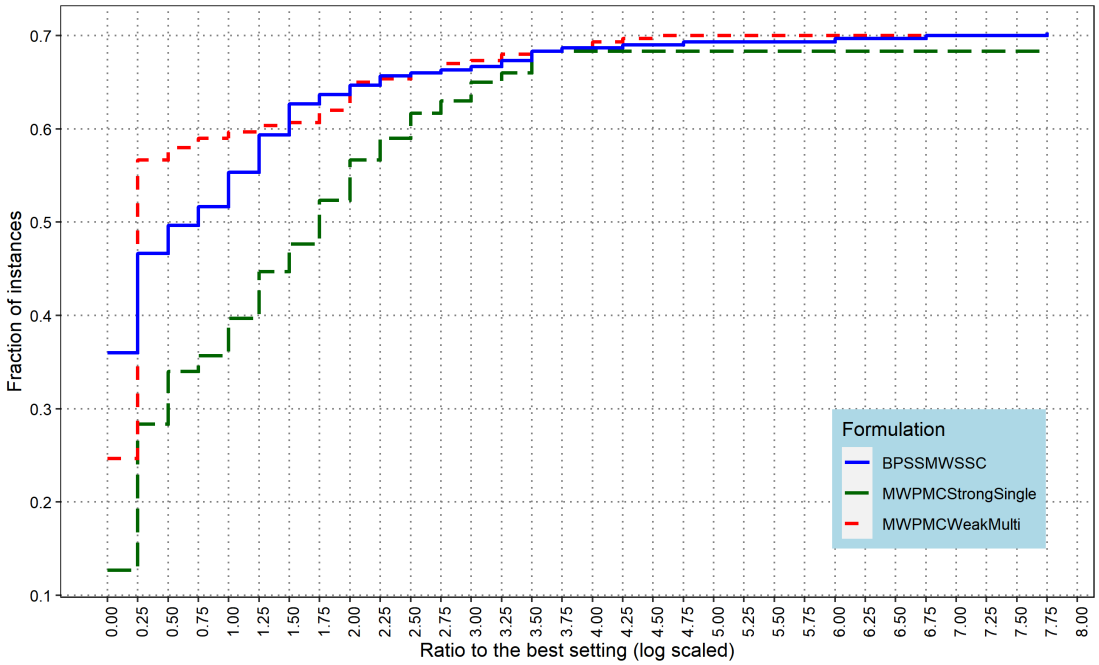

Figure 7.3. Performance profiles of the best methods.

Further results are reported in Tables 7.8 and 7.9, which contains time averages for each instance group. The raw tables for CPLEX and B&P runs are given in Appendix A.1 and Appendix A.2, respectively. The smallest instances with 20 vertices are solved in less than a second by both CPLEX formulations and B&P algorithm according to Table 7.8. One can also observe that Stable-Set-Based MWSSC algorithm spends much more time than both CPLEX formulations on the large instances that have a small graph density. However, as the graph density increases, the performance of Stable-Set-Based-MWSSC exceeds CPLEX formulations in the large instances.

Table 7.8. The effect of graph density on the best algorithms.

$\|V(G)\|$	Formulation	$\Gamma = 0.2$	$\Gamma = 0.5$	$\Gamma = 0.8$	Avg.
20	MWPMCStrongSingle	0	0.09	0.51	0.2
	MWPMCWeakMulti	0.01	0.04	0.11	0.05
	Stable-Set-Based-MWSSC	0	0.01	0.04	0.02
	Avg.	0	0.05	0.22	0.09
40	MWPMCStrongSingle	0.06	120.13	519.56	213.25
	MWPMCWeakMulti	0.02	38.86	488.5	175.79
	Stable-Set-Based-MWSSC	0.15	439.72	504.19	314.69
	Avg.	0.08	199.57	504.08	234.58
60	MWPMCStrongSingle	4.17	734.45	1186.29	641.64
	MWPMCWeakMulti	1.38	600.6	1112.06	571.35
	Stable-Set-Based-MWSSC	191.92	744.25	963.31	633.16
	Avg.	65.82	693.1	1087.22	615.38
80	MWPMCStrongSingle	211.63	1359.66	1800.02	1123.77
	MWPMCWeakMulti	42.24	1138.92	1800.45	993.87
	Stable-Set-Based-MWSSC	459.71	987.71	1385.67	944.36
	Avg.	237.86	1162.1	1662.05	1020.67
100	MWPMCStrongSingle	572.16	1369.29	1805.29	1248.91
	MWPMCWeakMulti	488.12	1708.02	1800.5	1332.21
	Stable-Set-Based-MWSSC	624.28	1383.46	1614.9	1207.55
	Avg.	561.52	1486.92	1740.23	1262.89
Overall Avg.		173.06	708.35	998.76	626.72

Table 7.9 displays the relationship between the performance and the conflict graph density. According to the table, the performance of all algorithm increases as the conflict graphs gets denser. Moreover, the performance of Stable-Set-Based-MWSSC gets even better than both CPLEX formulations. Finally, according to the rightmost column that shows the overall average of each row, Stable-Set-Based-MWSSC shows the best performance on the average while solving the largest instances.

Table 7.9. The effect of conflict graph density on the best algorithms.

$\|V(G)\|$	Formulation	$\Omega = 0.2$	$\Omega = 0.4$	$\Omega = 0.6$	$\Omega = 0.8$	Avg.
20	MWPMCStrongSingle	0.29	0.45	0.06	0.02	0.2
	MWPMCWeakMulti	0.08	0.08	0.03	0.02	0.05
	Stable-Set-Based-MWSSC	0.03	0.03	0.01	0	0.02
	Avg.	0.13	0.18	0.03	0.01	0.09
40	MWPMCStrongSingle	751.27	85.08	15.31	1.34	213.25
	MWPMCWeakMulti	649.22	38.8	13.74	1.41	175.79
	Stable-Set-Based-MWSSC	1181.8	75.25	1.51	0.19	314.69
	Avg.	860.76	66.38	10.19	0.98	234.58
60	MWPMCStrongSingle	1204.96	915.67	438.37	7.54	641.64
	MWPMCWeakMulti	1202.81	768.77	312.05	1.76	571.35
	Stable-Set-Based-MWSSC	1455.41	986.27	86.84	4.11	633.16
	Avg.	1287.73	890.24	279.09	4.47	615.38
80	MWPMCStrongSingle	1464.69	1215.88	1200.34	614.18	1123.77
	MWPMCWeakMulti	1251.63	1205.61	916.43	601.83	993.87
	Stable-Set-Based-MWSSC	1800	1211.82	711.78	53.84	944.36
	Avg.	1505.44	1211.1	942.85	423.28	1020.67
100	MWPMCStrongSingle	1800.02	1359.25	1200.93	635.45	1248.91
	MWPMCWeakMulti	1800.85	1250.35	1200.95	1076.69	1332.21
	Stable-Set-Based-MWSSC	1800.01	1424.07	1207.6	398.51	1207.55
	Avg.	1800.29	1344.56	1203.16	703.55	1262.89
Overall Avg.		1090.87	702.49	487.06	226.46	626.72

7.2.4. Branch-and-Bound Algorithm Results

The B&B algorithm developed for MWMC is also tested over the same test set. For all instances, both the following MWMC and MWSSP formulations are solved by CPLEX in multi-threaded mode.

$$\text{MWMC:} \quad \max \quad \mathbf{w}^T\mathbf{x} \tag{7.2}$$
$$\text{s.t.} \quad \mathbf{B}_G\mathbf{x} \leq \bar{\mathbf{1}}_{|V(G)|}, \tag{7.3}$$
$$\mathbf{B}_C^T\mathbf{x} \leq \bar{\mathbf{1}}_{|E(C)|}, \tag{7.4}$$
$$\mathbf{x} \in \mathbb{B}^{|E(G)|}. \tag{7.5}$$

$$\text{MWSSP:} \quad \max \quad \mathbf{w}^T\mathbf{x}, \tag{7.6}$$
$$\text{s.t.} \quad \mathbf{B}_{\hat{C}}^T\mathbf{x} \leq \bar{\mathbf{1}}_{|E(\hat{C})|}, \tag{7.7}$$
$$\mathbf{x} \in \mathbb{B}^{|E(G)|}. \tag{7.8}$$

The B&B algorithm is also run on the same instances. Table 7.10 shows the total numbers for each stopping situation. While the B&B algorithm solves 215 instances optimally, the numbers for MWMC and MWSSP are 161 and 153 respectively.

All 153 instances, which CPLEX-MWSSC formulation returns an optimal solution, are solved optimally by other alternatives. For the remaining instances, at least a feasible solution is found. According to the number of instances solved optimally, BB-MWMC algorithm is more efficient.

Table 7.10. MWMC results summary.

Algorithm	Optimal	Feasible	Total
BB-MWMC	215	85	300
CPLEX-MWMC	161	139	300
CPLEX-MWSSC	153	147	300

Figure 7.4 displays the performance profiles of formulations run on CPLEX and B&B algorithm developed for solving MWMC. It clearly indicates that B&B algorithm works much more efficiently than CPLEX in the test set. In almost 70% of the instances, B&B algorithm shows the best performance. Additionally, MWMC formulation shows a better performance than MWSSP formulation. In conclusion, using B&B algorithm to solve subproblems increases the performance of B&P algorithm, according to these results.

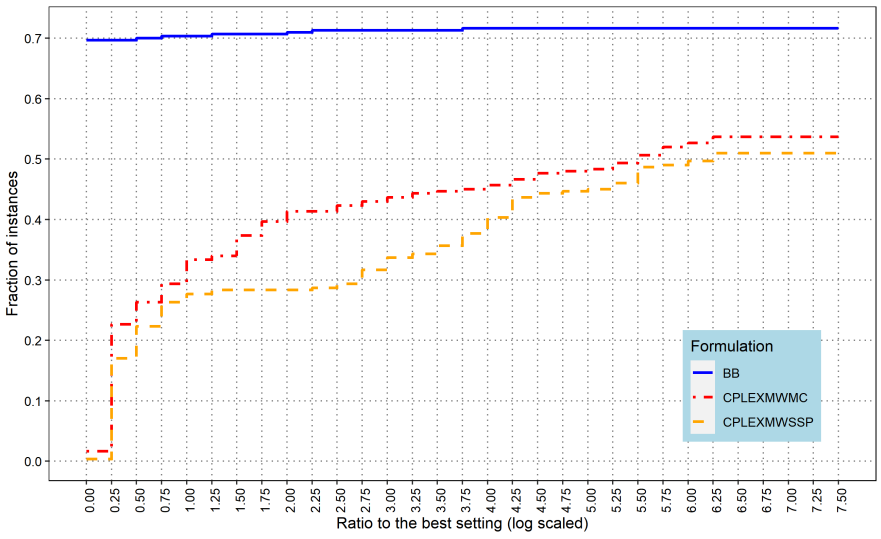

Figure 7.4. Performance profiles of MWMC algorithms.

Tables 7.11 and 7.12 show the average runtime for each instance configuration for the B&B algorithm (BB-MWMC) and each formulation (CP-MWMC, CP-MWSSP) considering the graph and conflict graph densities, respectively. The complete set of test results are contained in Appendix A.3. According to Table 7.11, B&B algorithm shows better performance than CPLEX formulations in all size-graph density combinations. There exists a single counterexample, which is the instances with 60 vertices and 0.2 graph density. Finally, the size of the graph and its density increases the difficulty of the instances.

Table 7.11. The effect of graph density on MWMC algorithms.

$\|V(G)\|$	Formulation	$\Gamma = 0.2$	$\Gamma = 0.5$	$\Gamma = 0.8$	Avg.
20	BB-MWMC	0	0.01	0.03	0.01
	CPLEX-MWMC	0.01	0.07	0.29	0.12
	CPLEX-MWSSC	0.01	0.08	0.42	0.17
	Avg.	0.01	0.05	0.25	0.1
40	BB-MWMC	0.2	186.2	501.55	229.32
	CPLEX-MWMC	0.45	467.54	947.93	471.97
	CPLEX-MWSSC	0.62	499.49	952	484.04
	Avg.	0.42	384.41	800.49	395.11
60	BB-MWMC	226.56	729.17	967.34	641.02
	CPLEX-MWMC	58.47	1186.34	1800.91	1015.24
	CPLEX-MWSSC	93.71	1205.63	1800.17	1033.17
	Avg.	126.24	1040.38	1522.81	896.48
80	BB-MWMC	459.28	981.27	1382.74	941.09
	CPLEX-MWMC	625.44	1800.17	1802	1409.2
	CPLEX-MWSSC	911.55	1800.16	1800.39	1504.03
	Avg.	665.42	1527.2	1661.71	1284.78
100	BB-MWMC	616.01	1376.87	1558.78	1183.88
	CPLEX-MWMC	1178.2	1802.06	1800.92	1593.73
	CPLEX-MWSSC	1362.21	1800.42	1802.93	1655.19
	Avg.	1052.14	1659.78	1720.88	1477.6
Overall Avg.		368.85	922.37	1141.23	810.81

Table 7.12 shows the performance of each MWMC solution method in each group of conflict graph densities. The results supports the conclusions derived from the previous table. In each group of instance, B&B algorithm shows clearly better performance than IP formulations. Moreover, as the conflict graph density increases, the difference between B&B algorithm and IP formulations gets even larger.

According to Table 7.12, B&B algorithm spends only around 49 seconds to solve instances with 80 vertices and 0.8 conflict graph density on the average. On the other hand, this value is around 1200 seconds on the average for both formulations. The only exception to this conclusion is the instances with 60 vertices and 0.2 conflict graph density. In conclusion, the proposed B&B algorithm is a strong alternative to CPLEX according to these results.

Table 7.12. The effect of conflict graph density on MWMC algorithms.

$\|V(G)\|$	Algorithm	$\Omega = 0.2$	$\Omega = 0.4$	$\Omega = 0.6$	$\Omega = 0.8$	Avg.
20	BB-MWMC	0.02	0.03	0.01	0	0.01
	CPLEX-MWMC	0.11	0.27	0.07	0.05	0.12
	CPLEX-MWSSC	0.13	0.35	0.15	0.05	0.17
	Avg.	0.09	0.21	0.08	0.03	0.1
40	BB-MWMC	843.06	72.59	1.43	0.18	229.32
	CPLEX-MWMC	1195.94	619.07	71.56	1.33	471.97
	CPLEX-MWSSC	1204.6	653.56	69.41	8.58	484.04
	Avg.	1081.2	448.41	47.47	3.36	395.11
60	BB-MWMC	1501.62	966.58	92.07	3.82	641.02
	CPLEX-MWMC	1276.87	1204.01	976.93	603.16	1015.24
	CPLEX-MWSSC	1316.81	1206.53	992.48	616.85	1033.17
	Avg.	1365.1	1125.71	687.16	407.94	896.48
80	BB-MWMC	1800	1211.29	703.64	49.44	941.09
	CPLEX-MWMC	1800.22	1419.56	1213.57	1203.47	1409.2
	CPLEX-MWSSC	1800.19	1758.66	1250.77	1206.51	1504.03
	Avg.	1800.13	1463.17	1055.99	819.8	1284.78
100	BB-MWMC	1800	1413.41	1207.25	314.87	1183.88
	CPLEX-MWMC	1800.31	1800.37	1571.3	1202.92	1593.73
	CPLEX-MWSSC	1800.53	1800.22	1801.57	1218.43	1655.19
	Avg.	1800.28	1671.33	1526.71	912.08	1477.6
Overall Avg.		1209.36	941.77	663.48	428.64	810.81

8. CONCLUSION

In this thesis, we develop Branch-and-Price algorithms for the Maximum Weight Perfect Matching Problem with Conflict Constraints. To this end, we first discuss BIP formulations of MWPMC. We address the existing formulations and propose novel formulations based on the idea of partitioning the conflict graph. Then, we suggest five Dantzig-Wolfe reformulations on which B&P algorithms are implemented. Finally, a novel B&B algorithm for MWMC, which is used as a subproblem solver in one of the B&P algorithms, is shown.

Over a large test set that contains 300 instances, we run tests to observe the performance of the formulations and developed algorithms. Firstly, we collect results by solving the instances using the proposed formulations on CPLEX solver. Then, we determine the best performing formulations. We also collect results by solving a subset of instances with the proposed B&P algorithms. Finally, the best performing B&P algorithm is compared with CPLEX results over the complete instance set. The results show that the best performing algorithm provides satisfactory results on large instances having a large conflict density.

The same test set is utilized to test B&B algorithm solving MWMC. The algorithm's performance is compared to CPLEX results, which are generated by solving MWMC and MWSSP formulations. According to these results, the B&B algorithm is more effective than CPLEX on this problem. Hence, it is suitable to solve the subproblems in column generation iterations.

While designing the algorithms, we assume that the graphs and subgraphs belong to arbitrary graph classes. Hence, the proposed algorithms solve $\mathcal{NP}$-hard problems in each column generation iteration, which reduces the efficiency of the algorithms. Future research may be done for MWPMC on specific graph classes to develop more efficient B&P algorithms.

Finally, B&P algorithm has a great number of studies that show tricks to increase the efficiency. We discuss a subset of these approaches and use some of the ideas in our implementations. Future research may be focused on other B&P-specific aspects which may result in better results. Additionally, one may focus on the introduction of valid inequalities for MWPMC and extend the proposed algorithms to Branch-Cut-and-Price strategies. Odd cycle inequalities is an example of such inequalities that can be utilized for matchings [2].

REFERENCES

1. Ahuja, R. K., T. L. Magnanti and J. B. Orlin, *Network Flows: Theory, Algorithms, and Applications*, Prentice Hall, Upper Saddle River, 1993.

2. Lovász, L. and M. D. Plummer, *Matching Theory*, Elsevier, Amsterdam, 1986.

3. Darmann, A., U. Pferschy, J. Schauer and G. J. Woeginger, "Paths, Trees and Matchings Under Disjunctive Constraints", *Discrete Applied Mathematics*, Vol. 159, pp. 1726–1735, 2011.

4. Barnhart, C., E. L. Johnson, G. L. Nemhauser, M. W. P. Savelsbergh and P. H. Vance, "Branch-and-Price: Column Generation for Solving Huge Integer Programs", *Operations Research*, Vol. 46, pp. 316–329, 1998.

5. Friberg, D., *An Implementation of the Branch-and-Price Algorithm Applied to Opportunistic Maintenance Planning*, Master's Thesis, Chalmers University of Technology, 2015.

6. Warrier, D., W. E. Wilhelm, J. S. Warren and I. V. Hicks, "A Branch-and-Price Approach for the Maximum Weight Independent Set Problem", *Networks*, Vol. 46, pp. 198–209, 2005.

7. Golumbic, M. C., *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, New York, 1980.

8. Şuvak, Z., *Network Flows with Conflict Constraints*, Ph.D. Thesis, Boğaziçi University, 2019.

9. Cunningham, W. H. and A. B. Marsh, "A Primal Algorithm for Optimum Matching", M. Balinski and A. Hoffman (Editors), *Mathematical Programming Studies*, Vol. 8, pp. 50–72, Springer, Berlin, Heidelberg, 1978.

10. Garey, M. R. and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman and Company, San Francisco, 1979.

11. Öncan, T., R. Zhang and A. P. Punnen, "The Minimum Cost Perfect Matching Problem with Conflict Pair Constraints", *Computers & Operations Research*, Vol. 40, No. 4, pp. 920–930, 2013.

12. Öncan, T. and İ. K. Altınel, "A Branch-and-Bound Algorithm for the Minimum Cost Bipartite Perfect Matching Problem with Conflict Pair Constraints", *Electronic Notes in Discrete Mathematics*, Vol. 64, pp. 5–14, 2018.

13. Öncan, T., Z. Şuvak, M. H. Akyüz and İ. K. Altınel, "Assignment Problem with Conflicts", *Computers and Operations Research*, Vol. 111, pp. 214–229, 2019.

14. Öncan, T., M. H. Akyüz and İ. K. Altınel, "A Branch-and-Bound Algorithm for the Maximum Weight Perfect Matching Problem with Conflicting Edge Pairs.", B. Darties and M. Poss (Editors), *Network Optimization INOC 2019: 9th International Network Optimization Conference*, pp. 31–36, Avignon, France, June 12-14, 2019, OpenProceedings, 2019.

15. Gusfield, D., *Integer Linear Programming in Computational and Systems Biology: An Entry-Level Text and Course*, Cambridge University Press, Cambridge, 2019.

16. Razafindrazaka, F. H., U. Reitebuch and K. Polthier, "Perfect Matching Quad Layouts for Manifold Meshes", *Computer Graphics Forum*, Vol. 34, pp. 219–228, 2015.

17. Yamada, T., S. Kataoka and K. Watanabe, "Heuristic and Exact Algorithms for the Disjunctively Constrained Knapsack Problem", *Information Processing Society of Japan Journal*, Vol. 43, pp. 2864–2870, 2002.

18. Hifi, M. and M. Michrafy, "A Reactive Local Search-Based Algorithm for the Disjunctively Constrained Knapsack Problem", *Journal of the Operational Research*

Society, Vol. 57, pp. 718–726, 2006.

19. Hifi, M. and M. Michrafy, "Reduction Strategies and Exact Algorithms for the Disjunctively Constrained Knapsack Problem", *Computers and Operations Research*, Vol. 34, pp. 2657–2673, 2007.

20. Pferschy, U. and J. Schauer, "The Knapsack Problem with Conflict Graphs", *Journal of Graph Algorithms and Applications*, Vol. 13, pp. 233–249, 2009.

21. Bettinelli, A., V. Cacchiani and E. Malaguti, "A Branch-and-Bound Algorithm for the Knapsack Problem with Conflict Graph", *INFORMS Journal on Computing*, Vol. 29, pp. 457–473, 2017.

22. Salem, M. B., R. Taktak, A. R. Mahjoub and H. Ben-Abdallah, "Optimization Algorithms for the Disjunctively Constrained Knapsack Problem", *Soft Computing*, Vol. 22, pp. 2025–2043, 2018.

23. Jansen, K., "An Approximation Scheme for Bin Packing with Conflicts", *Journal of Combinatorial Optimization*, Vol. 3, pp. 363–377, 1999.

24. Gendreau, M., G. Laporte and F. Semet, "Heuristics and Lower Bounds for the Bin Packing Problem with Conflicts", *Computers and Operations Research*, Vol. 31, pp. 347–358, 2004.

25. Muritiba, A. E., M. Iori, E. Malaguti and P. Toth, "Algorithms for the Bin Packing Problem with Conflicts", *INFORMS Journal on Computing*, Vol. 22, pp. 401–415, 2010.

26. Elhedhli, S., L. Li, M. Gzara and J. Naoum-Sawaya, "A Branch-and-Price Algorithm for the Bin Packing Problem with Conflicts", *INFORMS Journal on Computing*, Vol. 23, pp. 404–415, 2011.

27. Sadykov, R. and F. Vanderbeck, "Bin Packing with Conflicts: A Generic Branch-

and-Price Algorithm", *INFORMS Journal on Computing*, Vol. 25, pp. 244–255, 2013.

28. Ryan, D. M. and B. A. Foster, "An Integer Programming Approach to Scheduling", A. Wren (Editor), *Computer Scheduling of Public Transport: Urban Passenger Vehicle and Crew Scheduling*, pp. 269–280, North-Holland, Amsterdam, 1981.

29. Pferschy, U. and J. Schauer, "The Maximum Flow Problem with Disjunctive Constraints", *Journal of Combinatorial Optimization*, Vol. 26, pp. 109–119, 2013.

30. Şuvak, Z., İ. K. Altınel and N. Aras, "Exact Solution Algorithms for the Maximum Flow Problem with Additional Conflict Constraints", *European Journal of Operational Research*, Vol. 287, pp. 410–437, 2020.

31. Altınel, İ. K., N. Aras, Z. Şuvak and Z. C. Taşkın, "Minimum Cost Noncrossing Flow Problem on Layered Networks", *Discrete Applied Mathematics*, Vol. 261, pp. 2–21, 2019.

32. Şuvak, Z., İ. K. Altınel and N. Aras, "Minimum Cost Flow Problem with Conflicts", *Networks*, 2021, `https://doi.org/10.1002/net.22021`.

33. Desrosiers, J. and M. E. Lübbecke, "A Primer in Column Generation", G. Desaulniers, J. Desrosiers and M. M. Solomon (Editors), *Column Generation*, pp. 1–32, Springer-Verlag, New York, 2005.

34. Land, A. H. and A. G. Doig, "An Automatic Method of Solving Discrete Programming Problems", *Econometrica*, Vol. 28, pp. 497–520, 1960.

35. Vanderbeck, F., "Implementing Mixed Integer Column Generation", G. Desaulniers, J. Desrosiers and M. M. Solomon (Editors), *Column Generation*, pp. 331–358, Springer-Verlag, New York, 2005.

36. Dezso, B., A. Jüttner and P. Kovács, "LEMON - An Open Source C++ Graph

Template Library", *Electronic Notes in Theoretical Computer Science*, Vol. 264, pp. 23–45, 2011.

37. *ILOG CPLEX Optimization Studio 12.10.0 - IBM Documentation*, `https://www.ibm.com/docs/en/icos/12.10.0`, accessed in June 2021.

38. Grötschel, M., L. Lovász and A. Schrijver, "Stable Sets in Graphs", *Geometric Algorithms and Combinatorial Optimization*, Vol. 2, pp. 272–303, Springer, Berlin, Heidelberg, 1988.

39. Karypis, G. and V. Kumar, "A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs", *SIAM Journal of Scientific Computing*, Vol. 20, pp. 359–392, 1998.

40. Karypis, G. and V. Kumar, "METIS: A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices", *Computer Science & Engineering (CS&E) Technical Reports*, 1997, `https://hdl.handle.net/11299/215346`.

41. *Microsoft C/C++ Documentation*, `https://docs.microsoft.com/en-us/cpp/`, accessed in June 2021.

42. *MIP Dynamic Search Switch - IBM Documentation*, `https://www.ibm.com/docs/en/cofz/12.10.0?topic=parameters-mip-dynamic-search-switch`, accessed in June 2021.

43. *MIP Emphasis Switch - IBM Documentation*, `https://www.ibm.com/docs/en/cofz/12.10.0?topic=parameters-mip-emphasis-switch`, accessed in June 2021.

44. Dolan, E. D. and J. J. Moré, "Benchmarking Optimization Software with Performance Profiles", *Mathematical Programming*, Vol. 91, pp. 201–213, 2002.

APPENDIX A: RAW TEST RESULTS

The complete set of raw computational test results are given in the supplementary electronic files.

A.1. The Results for the Formulations Run on the Solver

Description: The table contains 9 columns, that respectively contains the name of the instance, the formulation that is run on the instance, the number of vertices in the instance, the graph and conflict graph densities, the return status, the time spent on the instance, the best objective function value found and the upper bound on termination.

File: MWPMC-MWSSC.xlsx

A.2. The Results for the Branch-and-Price Implementations

Description: All files contain 6 columns that are respectively indicating the name of the instance, the return status, the time used while solving the instance, the objective function value returned on termination, the upper bound and the optimality gap.

Files:

(i) Matching based MWPMC results: bp_mwpmc_matching.csv
(ii) Stable-set based MWPMC results: bp_mwpmc_stable_set.csv
(iii) Partitioning based MWPMC results: bp_mwpmc_part.csv
(iv) Stable-set based MWSSC results: bp_mwssc_stable_set.csv
(v) Partitioning-based MWSSC results: bp_mwssc_part.csv

A.3. The Results for Maximum Weight Matching Problem with Conflicts

Description: All files contain 6 columns that are respectively indicating the name of the instance, the return status, the time used while solving the instance, the objective function value returned on termination, the upper bound and the optimality gap.

Files:

(i) B&B algorithm results: bb_mwmc_lemon.csv
(ii) CPLEX with MWMC formulation results: cplex_mwmc_strong.csv
(iii) CPLEX with MWSSP formulation results: cplex_mwssp_strong.csv