A BRANCH, PRICE, AND CUT APPROACH TO SOLVING THE MAXIMUM WEIGHTED INDEPENDENT SET PROBLEM

A Dissertation

by

DEEPAK WARRIER

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

May 2007

Major Subject: Industrial Engineering

A BRANCH, PRICE, AND CUT APPROACH TO SOLVING THE MAXIMUM WEIGHTED INDEPENDENT SET PROBLEM

A Dissertation

by

DEEPAK WARRIER

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Approved by:

Chair of Committee,	Wilbert E. Wilhelm
Committee Members,	Illya V. Hicks
	Sergiy Butenko
	Jianer Chen
Head of Department,	Brett A. Peters

May 2007

Major Subject: Industrial Engineering

ABSTRACT

A Branch, Price, and Cut Approach to Solving the Maximum

Weighted Independent Set Problem. (May 2007)

Deepak Warrier, B.Tech., R.E.C Calicut;

M.S., Texas A&M University

Chair of Advisory Committee: Dr. Wilbert Wilhem

The maximum weight-independent set problem (MWISP) is one of the most well-known and well-studied NP-hard problems in the field of combinatorial optimization.

In the first part of the dissertation, I explore efficient branch-and-price (B&P) approaches to solve MWISP exactly. B&P is a useful integer-programming tool for solving NP-hard optimization problems. Specifically, I look at vertex- and edge-disjoint decompositions of the underlying graph. MWISP's on the resulting subgraphs are less challenging, on average, to solve. I use the B&P framework to solve MWISP on the original graph *G* using these specially constructed subproblems to generate columns. I demonstrate that vertex-disjoint partitioning scheme gives an effective approach for relatively sparse graphs. I also show that the edge-disjoint approach is less effective than the vertex-disjoint scheme because the associated DWD reformulation of the latter entails a slow rate of convergence.

In the second part of the dissertation, I address convergence properties associated with Dantzig-Wolfe Decomposition (DWD). I discuss prevalent methods for improving

the rate of convergence of DWD. I also implement specific methods in application to the edge-disjoint B&P scheme and show that these methods improve the rate of convergence.

In the third part of the dissertation, I focus on identifying new cut-generation methods within the B&P framework. Such methods have not been explored in the literature. I present two new methodologies for generating *generic* cutting planes within the B&P framework. These techniques are not limited to MWISP and can be used in general applications of B&P. The first methodology generates cuts by identifying faces (facets) of subproblem polytopes and lifting associated inequalities; the second methodology computes Lift-and-Project (L&P) cuts within B&P. I successfully demonstrate the feasibility of both approaches and present preliminary computational tests of each.

DEDICATION

To my family and friends for their patience, love and support

ACKNOWLEDGMENTS

I would like to acknowledge and express my gratitude towards my advisor Dr. Wilbert Wilhelm for his continuous support and guidance throughout my graduate study. He has been a source of encouragement and motivation through my endeavor. I would, also, like to extend a special thanks to Dr. Hicks. His input to my dissertation has been invaluable. I would also like to thank Dr. Butenko and Dr. Chen. They were a constant source of inspiration for me. I would also like to thank Judy for being helpful at all times.

TABLE OF CONTENTS

CHAPTER Page

CHAPTER Page

LIST OF FIGURES

LIST OF TABLES

CHAPTER I

INTRODUCTION

The maximum weight-independent set problem (MWISP) is one of the most well-known and well-studied NP-hard problems in combinatorial optimization. In the first part of this dissertation, we explore approaches based on branch-and-price (B&P) to solve MWISP exactly. In the second part of this dissertation, we address convergence properties of DWD and present specific techniques for improving the rate of convergence of DWD in application to our B&P scheme. In the final section of the dissertation, we focus on identifying new generic cut-generation methods for the B&P approach. These cut-generation methods are not limited to MWISP and can be used in general applications of the B&P.

In this chapter we introduce the notation that will be used in this dissertation. We also present some relevant background and a brief overview of the research. We begin section 1.1 by defining the notation and presenting some background on MWISP. We specify this research in section 1.2 in terms of its goals and objectives. We present our research motivation in section 1.3. We describe our method of approach in section 1.4. Finally, we conclude by presenting the organization of the dissertation in section 1.5.

This dissertation follows the format and style of Discrete Applied Mathematics.

1.1 Background

An independent set S in a graph is a subset of its vertices such that no two vertices in S are connected by an edge. Given a weighting of vertices, the *maximum weight independent set problem* (MWISP), which is NP-hard [58], is to prescribe an independent set of the graph that has maximum weight. The maximum weight-independent set problem (MWISP) is one of the most well-known and well-studied problems in the field of combinatorial optimization. It has many important applications, including combinatorial auctions [126], graph coloring [92], coding theory [88], geometric tiling [38], fault diagnosis [25], pattern recognition [70], molecular biology [57, 66, 93], and scheduling [72].

This dissertation deals with finite, simple undirected graphs. Graph $G = (V, E)$, comprises vertex set V and edge set E. An edge in E joining vertices $u, v \in V$ is denoted uv. The *neighbors* and *nonneighbors* of vertex $v \in V$ are $N(v) = \{u \in V : uv \in E\}$ and $\overline{N}(v) = (V \setminus (N(v) \bigcup \{v\})$, respectively. This notation is extended to a set $W \subseteq V$ by defining $N(W) = \bigcup_{v \in W} N(v) \setminus W$ and $\overline{N}(W) = (v \setminus N(W) \bigcup W)$. For any $W \subseteq V$, we denote the subgraph induced by W as $G[W]$; and for any $F \subseteq E$, we denote the subgraph induced by F as $G[F]$. For $V_1, V_2 \subset V$ the incidence of V_1 in V_2 is defined as $V_2(V_1) = V_2 \bigcap N(V_1)$. We use $S \subseteq V$ to denote an *independent set* of G and subgraph K to denote a *clique* of G (i.e., a complete subgraph of G).

The family of all independent sets is denoted S_G. For $S \subseteq V$ and $c \in R^n$, where $n = |V|$ we define $c(S) = \sum_{v_j \in S} c_j$. MWISP corresponds to determining $\{ \max\ c(S) \mid S \in S_G \}$. We assume $c_j \geq 0\ \forall\ v_j \in V$. MWISP can be formulated as a 0-1 integer program:

$$\max\ c\ x$$

$$s.t.\ \ A\ x\ \leq\ 1,\ x \text{ binary} \qquad (1)$$

where A: $m \times n$ edge-incidence matrix of G, $n = |V|$ and $m = |E|$.

In this formulation, a binary characteristic vector x represents a unique independent set from the set S_G. We are interested in the convex hull of these characteristic vectors, which is denoted $conv(S_G)$. The corresponding Linear Program or the fractional maximum weighted independent set problem (FMWISP) is obtained by relaxing the binary restrictions on x to $x \geq 0$. The corresponding convex hull is represented by L_G. A special case of MWISP is the *maximum independent set problem* (MISP) for which $c_j = 1\ \forall\ v_j \in V$. The corresponding linear relaxation will be referred to as MISLP.

1.2 Specifications of the research

This section describes the goals and objectives of this research.

1.2.1 Goal

This research has two primary goals. The first goal is to investigate approaches based on B&P to solve MWISP exactly. Specifically, our goal is to investigate two B&P

schemes: vertex- and edge-disjoint. Our aim is to determine the effectiveness of these two schemes in solving challenging instances of MWISP.

The second primary goal is to explore new generic approaches to generating cutting planes within the B&P framework. Such techniques have not been explored. We aim to provide a generic framework for generating cutting planes within the B&P approach. These techniques will be applied to MWISP in this dissertation but are not limited to it.

1.2.2 Research objectives

The objectives of this research are to present a rationale for using price-directed decomposition to solve MWISP, to investigate effective implementation techniques, and to conduct computational tests to identify strengths and weaknesses of the approach. The research comprises the following main tasks:

(1) Developing an effective vertex-disjoint B&P approach to solve MWISP exactly.

(2) Developing an effective edge-disjoint B&P approach to solve MWISP exactly.

(3) Developing insights into stabilizing column generation within B&P.

(4) Developing a generic cut-generation framework for B&P.

1.3 Motivation

MWISP is one of the most well-known and well-studied NP-hard problems in the field of combinatorial optimization. This research offers new approaches based on B&P to solve MWISP exactly. Specifically, the research presents two B&P schemes: vertex- and edge-disjoint. Our research evaluates the efficiency of these schemes

computationally, provides insights into the advantages and disadvantages of each, and offers guidelines for using each. Every solver developed for MWISP aims at being able to solve the entire spectrum of instances of MWISP but, often, the efficiency of a solver is limited to a certain range of instances. This research provides a scheme for embedding arbitrary MWISP solvers within our B&P framework. More importantly, this research shows that our B&P approach (using the embedded MWISP solver) is able to perform better on instances that were considered challenging for the embedded MWISP solver. Thus, our research provides a scheme for augmenting the performance of existing MWISP solvers.

Traditional methods for improving the rate of convergence of DWD involve parameters that are difficult to estimate a priori. Our research explores the convergence issues associated with DWD and attempts to provide insights into developing non-parametric methods to accelerate the rate of convergence of DWD.

Finally, this research lays the foundation for a generic framework for generating cutting planes within B&P. Cutting plane techniques are not used routinely within B&P because of the challenge in generating cutting planes in terms of the original decision variables. This research shows how to overcome this challenge and invoke cutting plane techniques within the B&P framework.

1.4 Method of approach

In this section, we discuss our method of approach.

Vertex disjoint B&P scheme. An approach that involves a DWD reformulation of the edge inequality formulation (1) of MWISP was developed by our research team and reported in [126]. It begins by partitioning the vertex set of the graph based on a vertex-disjoint partitioning. This approach employs a clustering heuristic-based partitioning scheme (METIS) and compares it to a chordal partitioning scheme. The METIS partitioning scheme aims at partitioning the vertex set equally among all partitions while attempting to minimize the number of edges that have ends in different sets [75-77]. The chordal partitioning scheme employs the procedure of Balas and Yu [13] to partition the vertex set such that each partition induces a chordal subgraph. This approach also employs two methods for managing the size of the restricted master problem (RMP) using either the basic edge-inequality formulation or a clique-based formulation. The latter aims at identifying a minimal set of cliques that cover all cross-edges (edges whose ends lie in different partitions) thus tightening the formulation and reducing degeneracy. Finally, two branching rules are explored–traditional variable-dichotomy branching and a special-purpose, branching on fractional-weighted cliques.

Edge disjoint B&P scheme. It has been shown that an arbitrary vertex-disjoint partitioning can be transformed to a corresponding edge-disjoint partitioning that yields a tighter bound [131]. We investigate whether this property can be extended to show that an arbitrary edge-disjoint partition always yields a tighter bound than an arbitrary vertex-disjoint partition having the same number of partitions. We employ a partitioning scheme based on branch decomposition [68] to create edge-disjoint subgraphs. We explore both edge partitioning and edge covering. The edge-disjoint formulation is

similar to the vertex cloning formulation described in [131] and involves invoking equality constraints among decision variables associated with a cloned vertex and each of its clones. Although these equalities tighten the formulation, they lead to higher orders of degeneracy and poor convergence. We adapt stabilization methods in an attempt to improve the rate of convergence. In addition, we invoke a clique cover of the cloned vertices and use these inequalities to tighten our formulation and to reduce degeneracy. We also explore problem-specific strategies to improve the rate of convergence and conduct a computational evaluation of these methods.

Stabilization techniques for B&P. The slow rate of convergence associated with the DWD reformulation affects the time spent at each node in the B&P tree and, hence, is critical to the efficiency of the approach. We develop insights into the instability issues that accompany column generation. Specifically, we discuss the convergence properties of DWD and prevalent techniques for improving the rate of convergence. We also present preliminary research towards developing a non-parametric approach to stabilizing DWD. Finally, we present techniques for improving the rate of convergence of the edge-disjoint B&P scheme.

Cut generation within B&P. Bounds obtained from the DWD relaxation are tighter than those given by the LP relaxation but are not tight enough to solve challenging instances effectively. In the second part of our research we focus on generating valid linear inequalities that can be incorporated in the B&P framework to tighten the formulation. Note that traditional generic cutting planes techniques– Gomory cutting planes and Lift-and-Project (L&P) cutting planes - are derived from the optimal simplex tableau, which

in a DWD reformulation will generate cutting planes in terms of the decision variables in the reformulated master problem. A cutting plane in terms of the master-problem variables can distort the subproblem structure. Hence, the challenge is to present techniques for generating cutting planes in terms of the original problem variables. This is the precise reason why cutting plane techniques are not used routinely in the B&P framework. We introduce a generic method for deriving cutting planes in the B&P framework in terms of the original problem variables. Although implemented specifically for MWISP, our approach will be useful in generic applications of B&P. We begin by identifying faces (facets) of the subproblem that are tight at the current DWD solution using a modification of the facet generation procedure (FGP) [102]. These valid inequalities, however, are of no use if incorporated in the master problem since they are implicitly invoked by the DWD reformulation, which optimizes over the convex hull of the feasible integer solutions to each subproblem. However, we show that these valid inequalities - when lifted across other subproblems- can potentially generate valid inequalities that cut off the current fractional solution in the master problem. Within the context of MWISP, this method identifies those facets of G that are obtained by lifting facets of polytopes associated with the subgraphs of G. However, we cannot guarantee that we will always be able to cut off the current fractional solution. In order to guarantee that a cut is always generated, we propose to use faces (facets) of the subproblem polytopes in conjunction with master problem inequalities in a Chvatal-Gomory (C-G) fashion to generate valid cutting planes. However a practical implementation of the C-G cut relies on identifying the C-G multipliers which is not

straightforward. We overcome this challenge by exploring the L&P technique and show how to invoke L&P cuts within a B&P framework. This is the basis of our cut generation scheme, which we evaluate in computational tests.

1.5 Organization of the dissertation

The dissertation is organized in eight chapters. Chapter II reviews literature relevant to this research. Chapter III addresses objective (1) and presents the vertex-disjoint scheme. Chapter IV addresses objective (2) and presents the edge-disjoint scheme. In Chapter V we focus on objective (3) and present techniques for improving the rate of convergence of DWD. Chapters VI and VII address objective (4) and each present a generic cut generation strategy for B&P. In Chapter VIII we present our conclusions and some recommendations for future research.

CHAPTER II

LITERATURE REVIEW

In this chapter we review literature relevant to this research. Section 2.1 presents some theoretical background of MWISP. A brief literature review of existing solvers for MWISP is given in section 2.2. In section 2.3 we present some background about B&P and review existing techniques towards improving the rate of convergence of DWD. Finally section 2.4 presents a brief review on cutting planes techniques.

2.1 MWISP

An important motivation for studying MWISP is that two classical combinatorial problems - set packing and set partitioning - can be transformed into MWISP on a corresponding intersection graph [100, 96]. One of the earliest attempts to explore the complete characterization of the convex hull of the independent set problem was by Padberg [100], who explored the facets of the set packing problem by equating it to MWISP on the underlying intersection graph. Padberg showed that maximal clique inequalities represent facets of $conv(S_G)$ and, further, that the only canonical inequalities (inequalities with 0-1 coefficients for its left-hand-side and a 1 for its right-hand-side) that are facets of $conv(S_G)$ correspond to maximal clique inequalities. Padberg also identified one other important family of facets of $conv(S_G)$: lifted odd-hole inequalities.

Nemhauser and Trotter [96] provided a complete characterization of the extreme points of $conv(L_G)$. They showed that, if x is an extreme point of $conv(L_G)$, then x_j equals $0, \frac{1}{2}, 1 \ \forall \ j \in V$. For $P \subseteq V$ let $x^P \in conv(L_G)$ be defined by $\{ x_j^P = \frac{1}{2}$ if $j \in P$; $x_j^P = 0$ else$\}$. If x^P as defined above is an extreme point in $conv(L_G)$ and $G[P]$ is connected, then x^P is said to be an elementary fractional extreme point. They showed that x^P is an elementary fractional extreme point IFF $G[P]$ contains an odd-cycle. They stated that x is an extreme point of $conv(L_G)$ IFF $x = x^0 + x^1 + \ldots + x^k$, where x^0 is an integer extreme point of $conv(L_G)$, $x^1 \ldots x^k$ are elementary fractional extreme points of $conv(L_G)$ and $x^0, x^1, \ldots, x^k$ are mutually disjoint. This implies that an arbitrary extreme point of $conv(L_G)$ can be represented uniquely as the sum of an integer extreme point and elementary fractional extreme points, and conversely, that any such sum of extreme points, which produce a feasible solution to MWISLP, produces an extreme point of $conv(L_G)$. They generalized the procedure of lifting odd-hole inequalities introduced by Padberg [100] and showed how to construct facets of $conv(S_G)$ from arbitrary facets associated with vertex-generated subgraphs of G. They made an important observation that the facets of $conv(S_G)$ can be divided into two distinct categories: those associated with subgraphs of G, which are obtained by lifting facets of polytopes associated with these subgraphs, and those uniquely associated with G. They emphasized that certain facets cannot be generated by simply lifting facets of subgraphs. In addition to the cliques and odd holes introduced by Padberg, they

introduced the odd anti-hole (an edge compliment of an odd hole). Finally, they showed that the facets obtained from lifting odd holes and cliques cut off all the fractional extreme points of $conv(L_G)$, noting at the same time that introducing these inequalities generally produces new fractional vertices.

Trotter [116] introduced other classes of facet-producing graphs called webs and anti-webs, subsuming cliques, odd holes and odd anti-holes. A sufficient local optimality condition for MWISP was presented by Nemhauser and Trotter [97]. They defined the concept of an augmenting set. Given $S \in S_G$, a vertex set $I \subseteq V \setminus S$ is called an augmenting subset to S if $I \in S_G$ and $c((S \bigcup I) \setminus S(I)) > c(S)$. They show that $S \in S_G$ is not an optimal independent set IFF $\exists$ some $I \subseteq V \setminus S$ which is augmenting to S. This implies, that given $S \in S_G$, we need to examine only those $I \in S_G$ for which $I \subseteq V \setminus S$ in order to improve upon S or verify its optimality. They also show that $S \in S_G$ is an optimal independent set in G IFF, for every maximal independent set $I \subseteq V \setminus S$, $S(I)$ is an optimum independent set in the bipartite subgraph $\hat{G}$ induced by $I \bigcup S(I)$. Further, if S is an optimal independent set in the subgraph induced by $S \bigcup N(S)$, then $S \subseteq S^*$, where S^* is an optimum independent set in G. They introduced the concept of persistency, showing that those variables that assume binary values in an optimum MWISLP solution retain the same values in an optimum solution. Suppose x^* is an optimal $(0, \frac{1}{2}, 1)$ - valued solution to MWISLP and $P = \{ v_j : x_j^* = 1 \}$; then, there exists an optimal independent set in G that contains P.

Chvatal [37] investigated the problem of finding a minimal description for $conv(S_G)$. They showed that $conv(S_G)$ can be represented by maximal clique and non-negativity inequalities IFF G is a perfect graph.

2.2 Solving MWISP

Different approaches for solving MISP exactly have been proposed. Explicit enumeration was proposed by Bron and Kerbosch [31]. B&B based approaches were explored by Balas and Yu [13] and, Carraghan and Pardalos [33], triggering the development of optimization methods for solving MWISP exactly [7, 11, 12, 26, 30, 32, 48, 67, 73, 89, 38–41, 47]. Nemhauser and Sigismondi [36] have proposed a cutting plane approach for MWISP. Mehrotra and Trick [92] proposed column generation to solve the *minimum coloring problem* employing MWISP subproblems.

2.3 B&P: convergence issues

Implementing column generation using Dantzig-Wolfe decomposition (DWD) within branch-and-bound (B&B) is referred to as branch-and-price (B&P) [16, 130]. Many NP-hard integer optimization problems have been approached using B&P [15, 16, 41, 42, 130].

Although a significant amount of research reports the successful use of B&P in various applications, it encounters difficulties in some applications. The bound obtained from DWD can be weak in comparison with the optimal objective value and lead to a

large B&B tree. The column-generation scheme could also be unstable, leading to a slow rate of convergence.

Recent research has focused on improving the rate of convergence of DWD, typically exploiting the dual space. A column in a primal linear program is equivalent to a row in the dual. Thus, column generation is equivalent to generating supporting hyperplanes to the epigraph of the piecewise linear dual function and solving the DWD is identical to solving the dual problem by Kelley's cutting plane method [78]. One reason for the slow rate of convergence observed in column generation is instability - upon adding a new cut in the dual space, the resulting dual solution can be far away from the current dual solution (irrespective of the fact that the current solution could be near-optimal or, in fact optimal) [40, 43]. Another component contributing to slow convergence is the tailing-off effect - the objective value improves rapidly early on but only slowly towards the end [40, 43]. Degeneracy, which is particularly significant for set partitioning problems, also affects convergence –column generation may require many iterations without improving the objective value [40, 43].

Many approaches work to stabilize the dual in an attempt to improve the rate of convergence. In the Boxstep method, optimization in the dual space is explicitly restricted to a trust region around the current dual solution. This trust region is redefined appropriately as the algorithm converges [90]. A trust region has been combined with a penalty function to prevent excessive dual oscillations [43]. In the analytic center cutting plane method (ACCPM) a central point relative to the current approximation of the dual function is used instead of the current optimal dual solution to generate columns

[61,62,63]. Smoothing approaches have also been proposed to capture the history of the column generation process by using some combination of all previously generated dual solutions along with the current dual optimal solution [129]. In a related approach, the next dual vector used to generate columns is obtained by taking a step away from the current dual solution in the direction of the best dual vector (the one corresponding to the best dual bound) obtained so far [129].

2.4 Cut generation techniques

Strong cutting plane methods form another, capable approach for solving 0-1 Integer programs. These methods aim at improving the current approximation of the integer convex hull by generating valid inequalities that cut off the current fractional solution of the linear programming relaxation. Dantzig, Fulkerson and Johnson were the first to employ a cutting plane approach when they solved the Traveling Salesman Problem (TSP). Gomory was the first to propose a cutting-plane approach as a generic solution procedure for solving pure 0-1 integer programs [64, 65]. Cutting plane algorithms can be broadly classified into two categories. The first is generic in the sense that the cutting planes generated don't rely on knowledge of the underlying combinatorial structure of the problem, whereas the second exploits the underlying combinatorial structure. Embedding cutting planes within B&B yields the branch-and-cut (B&C) approach.

A lot of interest has recently been regenerated in the area of generic cutting-plane methods. Gomory's cutting plane techniques, which were considered

computationally inefficient for some time, have recently been shown to be quite efficient if implemented within B&C and have been successfully incorporated within commercial software [5]. The disjunctive principle developed by Balas [2, 3, 4] has been explored further, leading to a new generic, cutting-plane method – lift and project (L&P), which is based on tightening the linear relaxation of an integer program by lifting the problem into a higher dimensional space where a tighter formulation is obtained. This higher dimension polyhedron, when projected back onto the original space, provides a tighter approximation of the integer convex hull [5, 6, 114]. L&P utilizes this higher dimension polyhedron to derive strong cutting planes for the original polyhedron [5, 6].

CHAPTER III

VERTEX-DISJOINT B&P SCHEME FOR MWISP

3.1 Introduction

This chapter presents the B&P scheme for solving MWISP on a given graph G that our research team developed and reported in [127]. It also notes the contributions made by the author of this dissertation. This scheme involves a vertex-disjoint decomposition of G and entails solving MWISP's on vertex-disjoint subgraphs of G within a B&P framework. MWISP's on the subgraphs are less challenging - in the average case - to solve than the MWISP on the original graph G. This study presents a rationale for using vertex-disjoint decompositions to solve MWISP exactly. This chapter comprises six sections. Section 3.2 presents the vertex-disjoint decomposition and the associated formulations. Sections 3.3, 3.4 and 3.5 present methods to deal with complexity of MWISP. Section 3.6 presents results of our computational tests and Section 3.7 presents our conclusions.

3.2 MWISP – formulations

The B&P approach uses the following edge-inequality based Integer Program (IP) to formulate MWISP:

$$Z^*_{MWISP} = \text{Max}\{\sum_{v \in V} w_v x_v : x \in Q\} \tag{3.1}$$

in which the inequalities corresponding to edges of G define Q:

$$Q = \{x \in B^{|V|} : x_u + x_v \leq 1, \forall uv \in E\}, \tag{3.2}$$

where $B^{|V|}$ denotes the set of binary vectors of dimension $|V|$ and binary variable $x_v = 1$ if vertex v is included in the independent set; else, $x_v = 0$. The set of feasible integral solutions to (3.2) represents the family of all independent sets in the graph G and is denoted S_G. We are interested in the convex hull of S_G, which is denoted by H_G (i.e., $H_G = conv(S_G)$). The corresponding linear relaxation of (3.1) is obtained by relaxing the binary restrictions on x_v to $0 \leq x_v \leq 1$. This LP is referred to as the fractional maximum weighted independent set problem (FMWISP). The corresponding convex hull is represented by L_G. In the next section we present formulations based on our vertex-disjoint decomposition.

3.2.1 Vertex-Disjoint formulations (VD)

We partition the vertex set of the graph $G = (V, E)$ into P parts $V_1,..,V_P$, yielding subgraph $G_p = G[V_p]$ with edge set $E_p = E(G_p)$ for each $p \in \{1,..,P\}$. The partition containing vertex v is denoted p_v. Edges of G whose end-points lie in disjoint partitions constitute set $\hat{E} = E \setminus \bigcup_{p=1}^{P} E_p$, which induces subgraph $G[\hat{E}]$. $\hat{V}$ denotes the vertex set of $G[\hat{E}]$. Based on this vertex-disjoint partitioning, MWISP can be formulated as:

$$Z^*_{MWISP} = \text{Max}\left\{\sum_{p=1}^{P}\sum_{v \in V_p} w_v x_v : x_u + x_v \leq 1, \forall uv \in \hat{E}, x^p \in Q_p, \forall p \in \{1,..,P\}\right\}, \tag{3.3}$$

where Q_p corresponds to edge-inequalities associated with G_p:

$$Q_p = \left\{ x \in B^{|V_p|} : x_u + x_v \le 1, \forall uv \in E_p \right\}. \tag{3.4}$$

Our B&P approach exploits the block-diagonal structure embedded within formulation (3.3):

$$Z^*_{MWISP} = \text{Max} \sum_{v \in V} w_v x_v = \text{Max} \sum_{p=1}^{P} w^p x^p$$

subject to

$$\begin{bmatrix} A_1 & A_2 & \dots & A_P \\ D_1 & 0 & \dots & 0 \\ 0 & D_2 & \dots & 0 \\ \dots & \dots & \ddots & \vdots \\ 0 & 0 & \dots & D_P \end{bmatrix} \begin{pmatrix} x^1 \\ x^2 \\ \vdots \\ x^P \end{pmatrix} \le 1 \tag{3.5}$$

$$x^p \in B^{|V_p|} \qquad \forall\ p \in \{1,..,P\},$$

where A_p is the matrix of coefficients in inequalities associated with edges $uv \in \hat{E}$, D_p is matrix of coefficients in inequalities associated with edges $uv \in E_p$, $x^p \in B^{|V_p|}$ is the vector of decision variables associated with vertices $v \in V_p$ and $w^p \in R^{|V_p|}$ is the corresponding vector of weights.

The set of integral solutions feasible with respect to E_p and $\hat{E}$ are denoted S_p and $S_{\hat{E}}$, respectively:

$$S_p = \{ x \in B^{|V|} : x_u + x_v \le 1, \forall (u,v) \in E_p \} \tag{3.6}$$

$$S_{\hat{E}} = \{ x \in B^{|V|} : x_u + x_v \le 1, \forall (u,v) \in \hat{E} \} \tag{3.7}$$

The corresponding integer convex hulls are denoted $H_p (= conv(S_p))$ and $H_{\hat{E}} (= conv(S_{\hat{E}}))$. The convex hull of the corresponding linear relaxations are denoted L_p and $L_{\hat{E}}$, respectively.

We reformulate MWISP by applying DWD [39] to the linear relaxation of (3.5). Within this scheme, a subproblem corresponds to each subgraph G_p, $p \in \{1,..,P\}$, while the master problem corresponds to the induced subgraph $G[\hat{E}]$. We solve MWISP's on the subgraphs to generate columns that populate the master problem. Each column in the master problem is hence associated with an extreme point in the corresponding subproblem polytope. The restricted master problem (RMP) involves a subset of such columns:

$$Z^*_{RMP_{VD}} = \text{Max} \sum_{p=1}^{P} \sum_{j \in J_p} \lambda_{jp} (w^p x^{jp}) \tag{3.8}$$

subject to

$$\sum_{p=1}^{P} \sum_{j \in J_p} \lambda_{jp} (A_p x^{jp}) \leq 1 \tag{3.9}$$

$$\sum_{j \in J_p} \lambda_{jp} = 1 \qquad \forall\ p \in \{1,..,P\} \tag{3.10}$$

$$\lambda_{jp} \geq 0 \qquad \forall\ p \in \{1,..,P\},\ j \in J_p, \tag{3.11}$$

where J_p is the set of integer extreme points of Q_p, $x^{jp} \in B^{|V_p|}$ is the vector defining extreme point $j \in J_p$, and λ_{jp} is the RMP decision variable corresponding to extreme point $j \in J_p$. MWISP subproblem p for $\{1,..,P\}$ is of the form:

$$Z_p^*(\alpha) = \text{Max}\left\{\left(w^p - A_p^T\alpha\right)x^p : x^p \in Q_p\right\}, \tag{3.12}$$

where $\alpha \in R^{|\hat{E}|}$ is the vector of dual variables associated with the rows of constraint set (3.9). A column corresponding to x^{jp} is deemed improving if $\left(w^p - A_p^T\alpha\right)x^{jp} - \beta_p > 0$, where β_p is the dual variable corresponding to the p^{th} convexity constraint (3.10).

B&P solves RMP to optimality at each node of the B&B tree using column generation. Variable fixing within child nodes of the B&B tree is enforced by inclusions (or exclusions) of the corresponding vertices in the subgraph G_{p_v}. The author of this dissertation implemented a generic B&P solver (see Appendix A) and adapted it for the above vertex-disjoint B&P scheme. In the next section we describe the partitioning scheme employed in this research.

3.3 Vertex-Disjoint partitioning

Our research team explore two alternative methods for partitioning the vertex set $V(G)$. The first method employs METIS [75-77], a clustering heuristic, to partition the vertex set $V(G)$ into a pre-specified number of subsets P. The author of this dissertation was involved in invoking METIS from the B&P solver. The METIS partitioning scheme (p1) aims at partitioning the vertex set $V(G)$ equally among all partitions while attempting to minimize the number of edges that have ends in different sets. We specify the number of partitions P primarily based upon the size and density of a subproblem that can typically be solved in an acceptable amount of time. The advantage of (p1) is that it allows the resulting density of $G[\hat{E}]$ to be controlled.

However, one disadvantage of (p1) is that the induced subgraphs do not have any special structure and, thus, there is no guarantee that they can be solved in polynomial time.

We compare this scheme to a chordal partitioning scheme which was developed by another member of our research team. The chordal partitioning scheme (p2) employs the procedure of Balas and Yu [13] to partition the vertex set such that each partition induces a chordal subgraph. The primary advantage of (p2) is that MWISP can be solved on each chordal subgraph in polynomial time. However (p2) results in a large $\left|\hat{E}\right|$ for a given number of partitions P; consequently, RMP is also large and requires substantial computational effort.

To solve MWISP on these subgraphs, another member of our research team adapted Carraghan-Pardalos algorithm [17]. In the next section we discuss methods for handling the associated RMP.

3.4 RMP

A large RMP affects the computational effectiveness of our B&P approach. Thus, the size of $\left|\hat{E}\right|$ is critical for our approach. Another issue observed, especially with more dense graphs, is degeneracy. Our research team employ two alternative methods to deal with RMP. The first method (m1) simply uses the constraints associated with edges in $\hat{E}$. The primary advantage of (m1) is its simplicity. However, it suffers from computational disadvantages due to the resulting size of RMP and its associated degeneracy. The second method (m2) aims at identifying a minimal set of cliques that

cover all edges in $\hat{E}$. This method employs a best-in greedy heuristic (devised and implemented by another member of our research team) to identify a set of cliques in $G[\hat{E}]$. This method entails solving a set-covering problem to select a minimal set of clique and edge inequalities that covers all edges in $\hat{E}$. This method offers the advantages of a tighter formulation due to the clique inequalities, thus providing a better bound in comparison with method (m1). It also reduces the order of degeneracy in comparison with (m1).

3.5 Branching

Our research team explored two alternative rules to branch upon obtaining a fractional RMP solution – traditional variable-dichotomy branching and a special-purpose branching on fractional-weighted cliques. The variable-dichotomy branching (b1) branches on the most fractional variable $x_v = \sum_{j \in J_{p_v}} \lambda_{jp_v} x^{jp_v}$, where $v \in V$, resulting in two child B&B nodes: one corresponding to vertex v being excluded ($x_v = 0$), and the other corresponding to vertex v being included ($x_v = 1$).

The second rule (b2) employed by our research team branches on the vertices of a clique in graph G. Weights are assigned to each vertex equivalent to the fractional value of its associated variable – specifically, vertex v is assigned a weight of $(0.5 - |x_v - 0.5|)$. This rule employs a greedy heuristic (implemented by another member of the research team) to identify a fractional-weighted clique K of large weight. We ensure that all vertices whose associated variables are fixed in the current B&B node are excluded

from clique K. Branching on clique K results in $|V(K)|+1$ child nodes: nodes $1,\cdots,|V(K)|$ correspond to a single new vertex $v \in V(K)$ being included ($x_v = 1$), while child node $|V(K)|+1$ corresponds to all vertices in K being excluded simultaneously ($x_v = 0 \ \forall v \in V(K)$). Note that since K contains only fractional vertices, it need not be maximal. Also, the current fractional solution need not violate K (i.e. K need not cut off the current fractional solution).

3.6 Computational results

This section describes the results of our computational tests. We conduct our tests on two sets of instances. The first set comprises unweighted instances from the Second DIMACS Implementation Challenge (we actually use the complements of the listed graphs). The second set comprises randomly generated π graphs. The parameter π defines the probability of an edge connecting two vertices in the graph ($0 \le \pi \le 1$). The weight of each vertex is generated from a discrete uniform distribution on the interval [1, M], with M = 1, 20, 40, 60, or 100. The M = 1 case corresponds to the unweighted case. For a given number of vertices $|V(G)|$ and a value of π, we generate 25 independent instances (each using a unique random number seed), comprising five subsets, each with a different value of M and each comprising five instances.

Table 1 compares performances of (m1) and (m2) using methods (p2) and (b2). The first six columns in Table 1 specify the instance: graph designation, the number of vertices, $|V|$; the number of edges, $|E|$; the % Density, Δ; the number of partitions (P);

and the corresponding $\left|\hat{E}\right|$. The last five columns give the method [(m1) or (m2)] and the results of each: the number of rows in RMP (RMP Rows); the number of B&B nodes required to find the optimal solution (B&B Nodes); the total number times RMP is solved (MP Sols); and the CPU run time for our B&P approach to prescribe an optimal solution (B&P Time). Results show that (m2) solves 7 of these 13 instances faster than (m1) (including three of the four most challenging instances), ties on five of the instances, and is substantially slower on only one instance (brock200-3). (m2) is at a disadvantage in terms of the additional time spent in identifying cliques. However, this disadvantage is overcome by the fact that (m2) typically yields a tighter model as indicated by comparing the number of B&B nodes with the corresponding number that (m1) achieves. (m2) also yields a smaller RMPs (see RMP Rows), requiring less computational effort. Overall, (m2) performs better than (m1) and, based on this comparison, we select (m2) as a default to use on other tests.

Table 2 compares performances of (p1) and (p2) using methods (m2) and (b2). Results show that (p2) solves 12 of the 13 instances faster than (p1) and essentially ties (p1) on the 13*th* instance (hamming8-2). Method (p1) typically results in larger $\left|\hat{E}\right|$, reflecting the fact that subproblems contain fewer edges. This affects the tightness of the bound, consequently, resulting in larger solution times. (p2) outperforms (p1) because it yields smaller RMPs, thus providing tighter bounds (see B&B Nodes). Based on this comparison, we prefer (p2) over (p1).

Table 3 compares the performances of (b1) and (b2) using methods (p2) and (m2). Results show that method (b2) is faster than (b1) on 10 of the 13 instances; it is significantly faster on the more challenging instances. Method (b2) incurs an overhead due to the time involved in finding cliques. However, (b2) makes up for this disadvantage since it requires much smaller B&B search-trees than (b1), on average (see B&B Nodes). We conclude that (b2) is superior because cliques provide better partitioning of the solution space. Based on these tests, we conclude that the (m2)-(p2)-(b2) combination is appropriate. We now evaluate (m2)-(p2)-(b2) combination further.

Table 4 compares the performance of our B&P approach to that of CPAA on the set of 13 DIMACS instances. The first four columns in Table 4 specify the instance: graph designation, $|V|$, $|E|$, and Δ. Columns 5-7 specify the number of partitions employed (P) and the resulting $|\hat{E}|$ and RMP Rows, respectively. To evaluate tightness of the formulation columns 8-10 list the upper bound corresponding to the optimal solution of RMP at the root node (Z_{LP}), the lower bound obtained from the heuristic (Z_H), and the optimal MWISP solution (Z_{IP}). Finally, columns 11-14 present relevant performance metrics: B&B Nodes; MP Sols; B&P Time; and CPAA Time, the CPU run time for CPAA to solve the instance. We evaluate the sensitivity of our B&P approach to the number of partitions by testing three different values of P on each instance. Our results show that the performance of our B&P approach is indeed sensitive to P and that it is more effective than CPAA on graphs with densities less than 40%. Our run times to solve these DIMACS instances are quite reasonable. CPAA can handle dense subgraphs

effectively; however, on large, sparse subgraphs, CPAA does not perform well. CPAA can efficiently handle sparse subgraphs with up to only 30-50 vertices. We select the value of P such that CPAA can efficiently solve the resulting subproblems. Thus, for sparse subgraphs we use larger values of P to make subgraphs smaller. However, as P increases, $\left|\hat{E}\right|$ also increases, weakening the bound provided by RMP. Smaller $\left|\hat{E}\right|$ is good in two ways: RMP requires less computational effort and subproblems contain more edge inequalities so that RMP provides tighter bounds.

Table 5 describes the random π graphs generated for testing; all graphs use $|V|$ = 100 and P = 4. Column 1 specifies the value of π. Columns 2-4, 5-7, and 8-10, give minimum, maximum, and average values (over five instances) for the resulting $\left|E\right|$, $\left|\hat{E}\right|$, and RMP Rows. Table 6 reports the results of test on these random graphs. Columns 1 in Table 6 specifies the value of π, and Column 2 specifies M. Columns 3-6 gives performance measures: Z^*_{LP}/Z^*_{IP}, Z^*_H/Z^*_{IP}, B&B Nodes, and RMP iterations. Columns 7-9, 10-12 specify the minimum, maximum, and average run times for our B&P approach and CPAA, respectively to solve a set of five random instances. As π increases, the upper bounds from the linear relaxations (Z^*_{LP}/Z^*_{IP} in column 3) as well as the lower bounds from the heuristic (Z^*_H/Z^*_{IP} values in column 4) degrade. The tightness of Z^*_{LP} the optimal solution at the root node, reduces as π increases because subgraphs contain fewer edges. Weaker bounds make denser problems more challenging (note B&B Nodes in column 5, RMP iterations in column 6 and run times in columns 7–9). Results in columns 5–12 show that, for a given π, the set of unweighted instances is

consistently more challenging than the set of related, weighted instances. Weighted instances (i.e., with M = 20, 40, 60, 100) have comparable run times for most values of π (exceptions are for π = 0.05 and for M = 100 with π = 0.10 and π = 0.15). Our B&P approach solves MWISP at the root node of the B&B tree in each instance with π = 0.01. CPAA failed to solve any instance with $0.01 \leq \pi \leq 0.10$, because each exceeded memory capacity (512 MB). Our B&P approach gives better run times for instances with $0.01 \leq \pi \leq 0.20$, but CPAA requires less run time on denser instances with $\pi \geq 0.30$.

3.7 Conclusions

In this chapter we present an approach developed by our research team for solving MWISP by utilizing a vertex-disjoint decomposition within a B&P framework. Tests conducted by our team indicate that this B&P approach is more effective on sparse graphs, which result in small RMPs. The tests on random π graphs also show that the unweighted maximum independent set problem is more challenging computationally than the corresponding weighted problem. Run time is sensitive to P, the number of vertex-disjoint partitions of the graph. The associated $\left|\hat{E}\right|$ is critical in defining the size of the master problem and the associated tightness of the bound. Overall, this B&P approach performs well on very sparse graphs, the category of instances that is most challenging for earlier approaches, including CPAA. Every approach developed for MWISP aims at being able to solve the entire spectrum of instances of MWISP but, often, the efficiency of a solver is limited to a certain range of instances. This research provides a scheme for embedding arbitrary MWISP solvers (CPAA in our case) within a

B&P framework. Further, this B&P approach (using the embedded MWISP solver) is able to perform better on instances that were considered challenging for the embedded MWISP solver. Thus, this research also provides a scheme for enhancing the performance of existing MWISP solvers such as CPAA.

Table 1
Comparison of methods (m1) and (m2)

Graph	$\|V\|$	$\|E\|$	Δ	P	$\|\hat{E}\|$	Method	RMP Rows	B&B Nodes	MP Sols.	B&P Time (sec.)
hamming8-2	256	1,024	3.1	20	626	(m1)	626	1	3	0.6
						(m2)	626	1	3	0.6
MANN_a9	45	72	7.3	5	26	(m1)	26	98	2,091	2.4
						(m2)	20	19	527	0.5
hamming6-2	64	192	9.5	8	96	(m1)	96	1	3	0.3
						(m2)	96	1	3	0.3
johnson8-4-4	70	560	23.2	3	240	(m1)	240	6	89	0.9
						(m2)	127	1	23	0.6
johnson16-2-4	120	1,680	23.5	8	1,082	(m1)	1,082	>3,500	>76,809	*
						(m2)	42	1	14	0.6
keller4	171	5,100	35.1	5	3,293	(m1)	3,293	21,067	405,073	4,557.1
						(m2)	1,995	12,523	307,029	1,812.2
hamming8-4	256	11,776	36.1	4	6,528	(m1)	6,528	1	6	4.4
						(m2)	3,707	1	12	6.1
brock200-3	200	7,852	39.5	2	3,463	(m1)	3,463	775	15,331	1,671.5
						(m2)	2,736	3,624	62,432	2,537.4
johnson8-2-4	28	168	44.4	8	137	(m1)	137	136	1,962	1.3
						(m2)	23	8	126	0.2
c-fat200-5	200	11,427	57.4	7	9,523	(m1)	9,523	45	1,321	86.1
						(m2)	9,393	33	1,238	86.3
p_hat300-1	300	33,917	75.6	2	16,580	(m1)	16,580	712	9,802	705.7
						(m2)	10,441	1,086	11,564	479.4
c-fat200-2	200	16,665	83.7	4	12,261	(m1)	12,261	17	233	20.9
						(m2)	11,406	8	127	19.2
c-fat200-1	200	18,366	92.3	2	8,999	(m1)	8,999	6	53	7.5
						(m2)	7,453	6	68	8.9

(m1) edge constraints only in master problem
(m2) clique constraints replace edge constraints in master problem
* exceeded memory capacity of 512 MB

Table 2 Comparison of methods (p1) and (p2)

Graph	$\|V\|$	$\|E\|$	Δ	P	Method	$\|\hat{E}\|$	RMP Rows	B&B Nodes	MP Sols.	B&P Time (sec.)
hamming8-2	256	1,024	3.1	20	(p1)	846	846	1	3	0.6
					(p2)	626	626	1	3	0.6
MANN_a9	45	72	7.3	5	(p1)	32	26	40	904	2.7
					(p2)	26	20	19	527	0.5
hamming6-2	64	192	9.5	8	(p1)	156	156	1	3	0.3
					(p2)	96	96	1	3	0.3
johnson8-4-4	70	560	23.2	3	(p1)	433	295	79	2,882	11.5
					(p2)	240	127	1	23	0.6
johnson16-2-4	120	1,680	23.5	8	(p1)	1,243	108	16	400	1.7
					(p2)	1,082	42	1	14	0.6
keller4	171	5,100	35.1	5	(p1)	4,499	3,226	>26,370	>853,215	*
					(p2)	3,293	1,995	12,523	307,029	1,812.2
hamming8-4	256	11,776	36.1	4	(p1)	10,650	7,434	>3,500	>38,126	*
					(p2)	6,528	3,707	1	12	6.1
brock200-3	200	7,852	39.5	2	(p1)	7,325	6,738	>9,334	>200,240	*
					(p2)	3,463	2,736	3,624	62,432	2,537.4
johnson8-2-4	28	168	44.4	8	(p1)	113	22	8	99	1.1
					(p2)	137	23	8	126	0.2
c-fat200-5	200	11,427	57.4	7	(p1)	11,201	11,077	>13	6,396	*
					(p2)	9,523	9,393	33	1,238	86.3
p_hat300-1	300	33,917	75.6	2	(p1)	31,511	24,833	>2,170	>84,660	*
					(p2)	16,580	10,441	1,086	11,564	479.4
c-fat200-2	200	16,665	83.7	4	(p1)	15,912	15,156	25	1,293	261.1
					(p2)	12,261	11,406	8	127	19.2
c-fat200-1	200	18,366	92.3	2	(p1)	16,696	14,504	90	2,047	158.5
					(p2)	8,999	7,453	6	68	8.9

(p1) partitioning the graph into chordal subgraphs
(p2) partitioning the graph using METIS
* exceeded memory capacity of 512 MB

Table 3
Comparison of methods (b1) and (b2)

Graph	$\|V\|$	$\|E\|$	Δ	P	$\|\hat{E}\|$	RMP Rows	Method	B&B Nodes	B&P Time (sec.)
hamming8-2	256	1,024	3.1	20	626	626	(b1)	1	0.7
						626	(b2)	1	0.6
MANN_a9	45	72	7.3	5	26	20	(b1)	13	0.6
						20	(b2)	19	0.5
hamming6-2	64	192	9.5	8	96	96	(b1)	1	0.2
						96	(b2)	1	0.3
johnson8-4-4	70	560	23.2	3	240	127	(b1)	1	0.6
						127	(b2)	1	0.6
johnson16-2-4	120	1,680	23.5	8	1,082	42	(b1)	*	*
						42	(b2)	1	0.6
keller4	171	5,100	35.1	5	3,293	1,995	(b1)	16,579	12,793.5
						1,995	(b2)	12,523	1,812.2
hamming8-4	256	11,776	36.1	4	6,528	3,707	(b1)	1	6.0
						3,707	(b2)	1	6.1
brock200-3	200	7,852	39.5	2	3,463	2,736	(b1)	>7,500	*
						2,736	(b2)	3,624	2,537.4
johnson8-2-4	28	168	44.4	8	137	23	(b1)	7	0.7
						23	(b2)	8	0.2
c-fat200-5	200	11,427	57.4	7	9,523	9,393	(b1)	51	293.2
						9,393	(b2)	33	86.3
p_hat300-1	300	33,917	75.6	2	16,580	10,441	(b1)	>2,000	*
						10,441	(b2)	1,086	479.4
c-fat200-2	200	16,665	83.7	4	12,261	11,406	(b1)	13	41.5
						11,406	(b2)	8	19.2
c-fat200-1	200	18,366	92.3	2	8,999	7,453	(b1)	9	9.5
						7,453	(b2)	6	8.9

(b1) branch on most fractional variable
(b2) branch on vertices (i.e., nodes) of a clique
* exceeded memory capacity of 512 MB

Table 4
Instances taken from the Second DIMACS Implementation Challenge solved using the (m2)-(p2)-(b2) combination of methods

Graph	$\|V\|$	$\|E\|$	Δ	P	$\|\hat{E}\|$	RMP Rows	Z_{LP}	Z_H	Z_{IP}	B&B Nodes	MP Sols.	B&P Time (sec.)	CPAA Time (sec.)
hamming8-2	256	1,024	3.1	11	493	493	128.0	128	128	1	3	1.0	*
hamming8-2	256	1,024	3.1	20	626	626	128.0	128	128	1	3	0.6	*
hamming8-2	256	1,024	3.1	24	716	716	128.0	128	128	1	3	0.6	*
MANN_a9	45	72	7.3	5	26	20	18.0	16	16	19	527	0.5	620.8
MANN_a9	45	72	7.3	6	29	22	18.0	16	16	25	687	0.5	620.8
MANN_a9	45	72	7.3	8	35	31	18.5	16	16	43	1,363	0.7	620.8
hamming6-2	64	192	9.5	4	64	64	32.0	32	32	1	3	0.3	*
hamming6-2	64	192	9.5	6	114	114	32.0	32	32	1	3	0.3	*
hamming6-2	64	192	9.5	8	96	96	32.0	32	32	1	3	0.3	*
johnson8-4-4	70	560	23.2	2	140	62	14.0	14	14	1	22	1.7	14.9
johnson8-4-4	70	560	23.2	3	240	127	14.8	14	14	1	23	0.6	14.9
johnson8-4-4	70	560	23.2	6	348	217	16.4	14	14	13	492	0.8	14.9
johnson16-2-4	120	1,680	23.5	6	1,088	15	8.0	8	8	1	9	0.6	*
johnson16-2-4	120	1,680	23.5	8	1,082	42	8.5	8	8	1	14	0.6	*
johnson16-2-4	120	1,680	23.5	10	1,234	128	10.5	8	8	11	335	0.9	*

* exceeded memory capacity of 512 MB

Table 4 continued

Graph	$\|V\|$	$\|E\|$	Δ	P	$\|\hat{E}\|$	RMP Rows	Z_{LP}	Z_H	Z_{IP}	B&B Nodes	MP Sols.	B&P Time (sec.)	CPAA Time (sec.)
keller4	171	5,100	35.1	4	3,003	1,853	17.8	8	11	14,456	329,556	1,934.2	3,075.4
keller4	171	5,100	35.1	5	3,293	1,995	18.1	8	11	12,523	307,029	1,812.2	3,075.4
keller4	171	5,100	35.1	8	3,744	2,554	20.7	8	11	24,690	694,846	12,831.5	3,075.4
hamming8-4	256	11,776	36.1	4	6,528	3,707	16.0	16	16	1	12	6.1	*
hamming8-4	256	11,776	36.1	5	7,707	4,505	20.8	16	16	440	21,373	536.5	*
hamming8-4	256	11,776	36.1	8	8,774	6,529	23.5	16	16	2,332	97,283	2,357.9	*
brock200-3	200	7,852	39.5	2	3,463	2,736	20.0	11	15	3,624	62,432	2,537.4	*
brock200-3	200	7,852	39.5	3	4,709	3,964	24.0	11	--	>10,024	>50,049	>14,400	*
johnson8-2-4	28	168	44.4	4	100	12	4.0	4	4	1	7	0.53	0.0
johnson8-2-4	28	168	44.4	5	124	32	5.3	4	4	6	95	0.45	0.0
johnson8-2-4	28	168	44.4	8	137	23	5.0	4	4	8	126	0.23	0.0

* exceeded memory capacity of 512 MB

Table 4 continued

Graph	$\|V\|$	$\|E\|$	Δ	P	$\|\hat{E}\|$	RMP Rows	Z_{LP}	Z_H	Z_{IP}	B&B Nodes	MP Sols.	B&P Time (sec.)	CPAA Time (sec.)
c-fat200-5	200	11,427	57.4	4	8,118	7,985	66.7	58	58	33	1,328	157.8	41.4
c-fat200-5	200	11,427	57.4	7	9,523	9,393	66.7	58	58	33	1,238	86.3	41.4
c-fat200-5	200	11,427	57.4	8	9,787	9,655	66.7	58	58	33	1,599	112.1	41.4
p_hat300-1	300	33,917	75.6	2	16,580	10,441	12.9	5	8	1,086	11,564	479.4	3.9
p_hat300-1	300	33,917	75.6	3	21,972	13,968	16.0	5	8	4,032	44,928	2,302.3	3.9
p_hat300-1	300	33,917	75.6	5	26,448	17,813	20.7	5	8	8,834	122,254	6,411.4	3.9
c-fat200-2	200	16,665	83.7	4	12,261	11,406	26.2	24	24	8	127	19.2	1.2
c-fat200-2	200	16,665	83.7	5	13,156	12,300	25.5	24	24	8	138	22.6	1.2
c-fat200-2	200	16,665	83.7	8	14,504	13,783	26.9	24	24	26	685	56.6	1.2
c-fat200-1	200	18,366	92.3	2	8,999	7,453	13.0	12	12	6	68	8.9	1.0
c-fat200-1	200	18,366	92.3	3	12,137	9,996	14.0	12	12	19	219	19.4	1.0
c-fat200-1	200	18,366	92.3	6	15.232	12,807	13.3	12	12	13	181	24.6	1.0

* exceeded memory capacity of 512 MB

Table 5
Randomly generated graphs

π	$\lvert E \rvert$			$\lvert \hat{E} \rvert$			RMP Rows		
	Min	Max	Avg	Min	Max	Avg	Min	Max	Avg
0.01	37	62	50.0	0	3	0.4	0	3	0.4
0.05	219	276	246.2	67	107	91.4	67	107	86.6
0.10	461	534	497.2	224	281	252.7	195	243	220.5
0.15	704	780	742.6	390	453	419.2	332	397	361.8
0.20	949	1,040	983.1	549	629	591.6	462	538	503.2
0.30	1,405	1,530	1,471.9	883	977	939.0	747	819	786.4
0.40	1,919	2,089	1,981.6	1,268	1,407	1,321.4	1,025	1,143	1,079.4
0.50	2,395	2,586	2,475.8	1,622	1,779	1,697.2	1,231	1,362	1,325.2

Table 6
Test results on randomly generated graphs

		Overall measures				B&P Run Times			CPAA Run Times		
π	M	Z^*_{LP} / Z^*_{IP}	Z_H / Z^*_{IP}	B&B Nodes	RMP Iterations	Min	Max	Avg	Min	Max	Avg
0.01	1	1.00	1.00	1.0	3.0	0.3	0.6	0.4	*	*	*
	20	1.00	0.91	1.0	3.0	0.2	0.5	0.3	*	*	*
	40	1.00	0.98	1.0	3.0	0.2	0.5	0.3	*	*	*
	60	1.00	0.98	1.0	3.0	0.2	0.5	0.3	*	*	*
	100	1.00	0.98	1.0	3.0	0.2	0.4	0.3	*	*	*
0.05	1	1.03	0.93	11.2	553.4	25.1	65.8	36.8	*	*	*
	20	1.01	0.94	6.2	350.8	3.5	27.9	10.2	*	*	*
	40	1.00	0.91	0.6	56.2	0.8	7.7	4.5	*	*	*
	60	1.00	0.93	1.2	74.6	0.3	7.3	3.3	*	*	*
	100	1.01	0.94	7.0	453.4	0.3	52.2	19.6	*	*	*
0.10	1	1.21	0.86	696.4	31,760.6	88.5	437.3	261.0	*	*	*
	20	1.06	0.92	22.6	1,348.6	13.8	46.3	27.1	*	*	*
	40	1.11	0.91	106.0	6,224.4	29.8	137.3	72.7	*	*	*
	60	1.10	0.93	72.8	4,459.2	27.7	97.7	55.3	*	*	*
	100	1.13	0.90	182.6	11,423.2	66.9	295.0	128.5	*	*	*
0.15	1	1.33	0.87	365.0	69,322.0	138.4	538.5	311.3	11,125.0	16,046.0	12,745.0
	20	1.24	0.91	384.6	17,263.0	29.7	135.9	84.1	1,688.6	5,281.4	3,100.2
	40	1.23	0.90	331.6	16,167.8	28.6	129.84	79.8	1,772.4	4,011.2	2,879.2
	60	1.24	0.89	247.2	11,392.4	44.0	76.9	59.2	1,992.0	4,626.9	2,998.8
	100	1.26	0.92	365.4	25,243.6	45.1	225.5	112.6	1,844.8	3,355.2	2,845.3
0.20	1	1.43	0.82	3,516.8	109,975.4	179.1	548.9	362.3	836.6	1,991.5	1,527.6
	20	1.30	0.90	403.8	15,594.6	35.7	83.2	63.9	245.4	1,032.8	570.6
	40	1.30	0.90	483.8	17,932.6	30.3	82.5	69.1	321.4	772.9	529.1
	60	1.31	0.91	447.0	17,645.2	46.3	83.3	67.4	212.6	1,141.9	437.8
	100	1.32	0.87	505.4	19,118.6	41.8	92.1	72.7	356.9	595.7	471.3
0.30	1	1.55	0.85	2,142.4	49,392.2	77.7	220.1	148.5	58.1	73.2	66.7
	20	1.43	0.88	351.2	9,881.2	25.2	45.2	34.4	15.2	30.3	20.6
	40	1.46	0.90	455.6	13,170.4	21.2	69.1	43.8	18.7	34.0	24.1
	60	1.46	0.86	687.8	18,199.4	38.7	122.9	59.6	19.6	40.9	28.7
	100	1.40	0.88	351.2	10,186.8	20.5	48.5	33.9	12.7	35.5	22.7

* exceeded memory capacity of 512 MB

Table 6 Continued

		Overall measures				B&P Run Times			CPAA Run Times		
π	M	Z^*_{LP} / Z^*_{IP}	Z_H / Z^*_{IP}	B&B Nodes	RMP Iterations	Min	Max	Avg	Min	Max	Avg
0.40	1	1.68	0.83	1,327.2	25,159.8	35.3	114.8	84.4	4.3	8.6	6.6
	20	1.57	0.86	444.6	9,500.2	19.5	43.9	33.8	2.5	3.7	3.1
	40	1.64	0.87	590.0	12,355.0	33.9	53.8	41.3	2.6	3.8	3.3
	60	1.51	0.87	353.2	7,780.8	19.3	37.1	27.2	3.2	4.4	3.8
	100	1.59	0.87	548.2	11,431.0	20.6	56.3	39.6	2.9	4.2	3.6
0.50	1	1.66	0.73	605.6	10,261.4	26.2	52.0	39.0	1.1	1.4	1.2
	20	1.68	0.78	377.0	6,637.4	18.2	35.7	25.1	0.7	1.0	0.8
	40	1.61	0.76	287.4	5,117.8	9.6	28.8	19.7	0.6	0.8	0.7
	60	1.56	0.75	227.6	4,100.2	12.9	20.1	16.3	0.6	0.8	0.7
	100	1.64	0.87	370.6	6,441.4	23.1	30.8	27.4	0.6	0.8	0.7

* exceeded memory capacity of 512 MB

CHAPTER IV

EDGE-DISJOINT B&P SCHEME FOR SOLVING MWISP

4.1 Introduction

It has been shown that an arbitrary vertex-disjoint partitioning can be transformed to a corresponding edge-disjoint partitioning that yields a tighter bound [131]. In the first half of this chapter we investigate whether this property can be extended further to show whether an arbitrary edge-disjoint partition yields a tighter bound than an arbitrary vertex-disjoint partition having the same number of partitions. We show that this is not guaranteed for an arbitrary partition. In the second half of the chapter we present a B&P scheme for solving MWISP based on an edge-decomposition of the original graph. We compare it with the vertex-disjoint scheme discussed in Chapter III. This study presents a rationale for using edge-based decompositions to solve MWISP exactly.

This chapter comprises seven sections. Section 4.2 presents the edge-disjoint decomposition and the associated formulations. Section 4.3 presents an analysis of bounds comparing arbitrary vertex and edge-disjoint decompositions. Sections 4.3, 4.4 and 4.5 present methods to manage the complexity of MWISP. Section 4.6 presents the results of our computational tests and section 4.7 gives our conclusions. In the next section we describe the edge-disjoint formulation. Relevant MWISP formulations from Chapter III are referenced and are not duplicated here.

4.2 MWISP – formulations

We refer to the edge-inequality based Integer Programming (IP) formulation for MWISP described in Chapter III. The vertex-disjoint formulation described in Chapter III is also referenced.

4.2.1 Edge-Disjoint formulation (ED)

We begin by partitioning the edge set of the graph $G = (V, E)$ into P parts $E_1, .., E_P$, defining subgraph $G_p = G[E_p]$ with edge set E_p for each $p \in \{1, .., P\}$. The partition containing edge uv is denoted p_{uv}. Vertices of G whose incident edges lie in more than one partition constitute set $\tilde{V}$, which induces subgraph $G[\tilde{V}]$. $\tilde{E}$ denotes the edge set of $G[\tilde{V}]$. For each $v \in \tilde{V}$ we define $\tilde{P}_v = \{p \in \{1, .., P\} \mid \exists u \in V \text{ with } uv \in E_p\}$, the set of partitions containing vertex v. A distinct decision variable is used to represent a vertex $v \in \tilde{V}$ in every partition $p \in \tilde{P}_v$ in which it appears. Thus, corresponding to each such vertex $v \in \tilde{V}$, we introduce $\left|\tilde{P}_v\right|$ decision variables $x_{v_k}, k \in \{1, .., \left|P_v\right|\}$ so that the edge-disjoint formulation has $\left|V \setminus \tilde{V}\right| + \left|\tilde{V}\right| \times \sum_{v \in \tilde{V}} \left|\tilde{P}_v\right|$ decision variables. The edge-disjoint formulation invokes equality constraints to equate the decision variables x_{v_k} that correspond to each vertex $v \in \tilde{V}$. Based on this edge-disjoint partitioning, MWISP can be expressed as:

$$Z^*_{MWISP} = \text{Max} \left\{ \begin{array}{l} \sum_{v \in V} w_v x_v + \sum_{v \in \tilde{V}} \sum_{k \in \{1, .., \left|\tilde{P}_v\right|, .\}} w_{v_k} x_{v_k} : \\ x_{v_k} - x_{v_{1,}} = 0, \forall k \in \{2, .., \left|\tilde{P}_v\right|\}, v \in \tilde{V}, x^p \in Q_p, \forall p \in \{1, .., P\} \end{array} \right\}, \quad (4.1)$$

where Q_p corresponds to edge-inequalities associated with G_p:

$$Q_p = \left\{ x \in B^{|V_p|} : x_u + x_v \leq 1, \forall uv \in E_p \right\}. \tag{4.2}$$

Similar to the vertex-disjoint formulation, the edge-disjoint formulation (4.1) has a block-diagonal structure, which is exploited by B&P:

$$Z^*_{MWISP} = \text{Max} \sum_{v \in V} w_v x_v = \text{Max} \sum_{p=1}^{P} w^p x^p$$

subject to

$$\begin{bmatrix} A_1 & A_2 & \dots & A_P \\ D_1 & 0 & \dots & 0 \\ 0 & D_2 & \dots & 0 \\ \dots & \dots & \ddots & \vdots \\ 0 & 0 & \dots & D_P \end{bmatrix} \begin{pmatrix} x^1 \\ x^2 \\ \vdots \\ x^P \end{pmatrix} \begin{pmatrix} = \\ \leq \\ \leq \\ \vdots \\ \leq \end{pmatrix} \begin{bmatrix} 0 \\ 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \tag{4.3}$$

$$x^p \in B^{|V_p|} \qquad \forall\ p \in \{1,..,P\},$$

where A_p is matrix of coefficients in equalities associated with vertices $v \in \tilde{V}$ in partition $p \in P$, D_p is matrix of coefficients in inequalities associated with edges $uv \in E_p$, $x^p \in B^{|V_p|}$ is the vector of decision variables representing vertices in $G[E_p]$, and $w^p \in R^{|V_p|}$ is the corresponding vector of weights.

The set of integral solutions feasible with respect to E_p is denoted S_p and the set of integral solutions feasible with respect to the equality constraints corresponding to each $v \in \tilde{V}$ is denoted $S_{\tilde{V}}$:

$$S_p = \left\{ x \in B^{|V \setminus \tilde{V}| + |\tilde{V}| \times \sum_v |P_v|} : x_u + x_{v_{1'}} \leq 1, \forall (u,v) \in E_p \right\} \tag{4.4}$$

and

$$S_{\tilde{v}} = \left\{ x \in B^{|V \setminus \tilde{V}| + |\tilde{V}| \times \sum_v |P_v|} : x_{v_k} - x_{v_{1'}} = 0, \forall k \in \{2,..,|\tilde{P}_v|\}, v \in \tilde{V} \right\}. \tag{4.5}$$

The corresponding integer convex hulls are denoted H_p (= $conv(S_p)$) and $H_{\tilde{V}}$ (= $conv(S_{\tilde{V}})$) and the convex hulls of the corresponding linear relaxations are denoted L_p and $L_{\tilde{V}}$, respectively.

We reformulate MWISP by applying DWD [39] to the linear relaxation of (4.3). Within this scheme, we have a subproblem corresponding to each subgraph G_p, $p \in \{1,..,P\}$. The master problem comprises the equality constraints corresponding to each $v \in \tilde{V}$. We solve MWISP's on the subgraphs to generate columns that populate the master-problem basis. Each column in the master problem is thus associated with an extreme point in the corresponding subproblem polytope. The restricted master problem (RMP) involves a subset of such columns:

$$Z^*_{RMP_{ED}} = \text{Max} \sum_{p=1}^{P} \sum_{j \in J_p} \lambda_{jp} (w^p x^{jp}) \tag{4.6}$$

subject to

$$\sum_{p=1}^{P} \sum_{j \in J_p} \lambda_{jp} (A_p x^{jp}) = 0 \tag{4.7}$$

$$\sum_{j \in J_p} \lambda_{jp} = 1 \qquad \forall \ p \in \{1,..,P\} \tag{4.8}$$

$$\lambda_{jp} \geq 0 \qquad \forall\ p \in \{1,..,P\},\ j \in J_p \tag{4.9}$$

where J_p is the subset of integer extreme points of Q_p that form columns in RMP, $x^{jp} \in B^{|V_p|}$ is the vector defining extreme point $j \in J_p$, and λ_{jp} is the RMP decision variable corresponding to extreme point $j \in J_p$. MWISP subproblem p for $\{1,..,P\}$ is of the form:

$$Z_p^*(\alpha) = \text{Max}\left\{\left(w^p - A_p^T\alpha\right)x^p : x^p \in Q_p\right\}, \tag{4.10}$$

where $\alpha \in R^{|\hat{E}|}$ is the vector of dual variables associated with the rows of constraint set (4.7). A column corresponding to x^{jp} is deemed improving if $\left(w^p - A_p^T\alpha\right)x^{jp} - \beta_p > 0$, where β_p is the dual variable corresponding to the p^{th} convexity constraint (4.8).

4.3 Bounds analysis

Sachdeva & Wilhelm [131] introduced vertex cloning to transform a given vertex-disjoint partition into an edge-disjoint partition. Vertex cloning involves duplicating vertices in $G[\hat{E}]$. Specifically, it replaces every edge-inequality $x_u + x_v \leq 1$ (where $uv \in \hat{E}$) in the master problem by an equality $x_v = x_c$ (vertex v in partition p_v is cloned as vertex c in partition p_u) and an inequality corresponding to clone x_c, $x_u + x_c \leq 1$ (associated with edge uc) in partition p_u. Cloning thus transforms a given vertex-disjoint partition into an edge-disjoint partition. They show that the bound obtained from the resulting edge-disjoint partitioning is not weaker than that associated with the corresponding vertex-disjoint partitioning. Their result thus implies that, an

arbitrary vertex disjoint partitioning can be transformed to an edge disjoint partitioning that yields a tighter bound. This leads us to the question of whether this result can be generalized to - 'does an arbitrary edge-disjoint partition yield a tighter bound than an arbitrary vertex-disjoint partition having the same number of partitions?' We investigate this question in this section. We first illustrate why a tightening is guaranteed with vertex-cloning. We then present insight into why tightening is not guaranteed with an arbitrary edge-disjoint partitioning and finally provide an example, which answers the question in the negative.

4.3.1 Bounds analysis: vertex-disjoint vs. vertex-cloning

In this section we illustrate vertex cloning and provide an insight that rationalizes the observed tightness. Figure 1 depicts an arbitrary graph with 6 vertices and 8 edges and an arbitrary vertex-disjoint partition of this graph into 2 partitions. Following Sachdeva and Wilhelm [131], Figure 2 shows the transformation of the vertex-disjoint partition to an edge-disjoint partition through vertex-cloning.

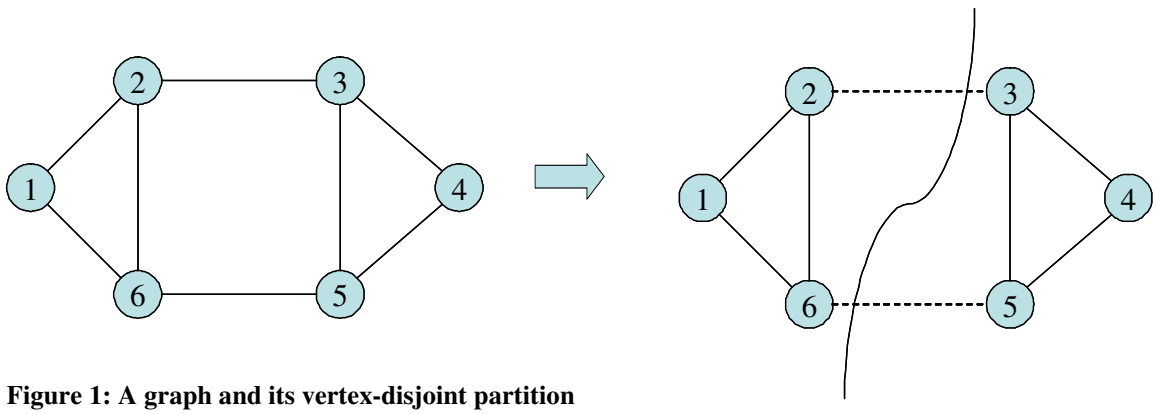


Figure 1: A graph and its vertex-disjoint partition

Note that, during this cloning transformation, the original vertex-disjoint subgraphs are augmented by adding more edges into them (in Figure 2 we add edges (2, 3) and (6, 5) into the subgraph on the left). Thus, the corresponding subproblem polytopes now more closely represent the polytope of the original MWISP. This is the primary reason that allows vertex cloning to provide tighter bounds. In addition, equalities invoked in the master problem ensure that the feasible region of the edge-disjoint formulation is

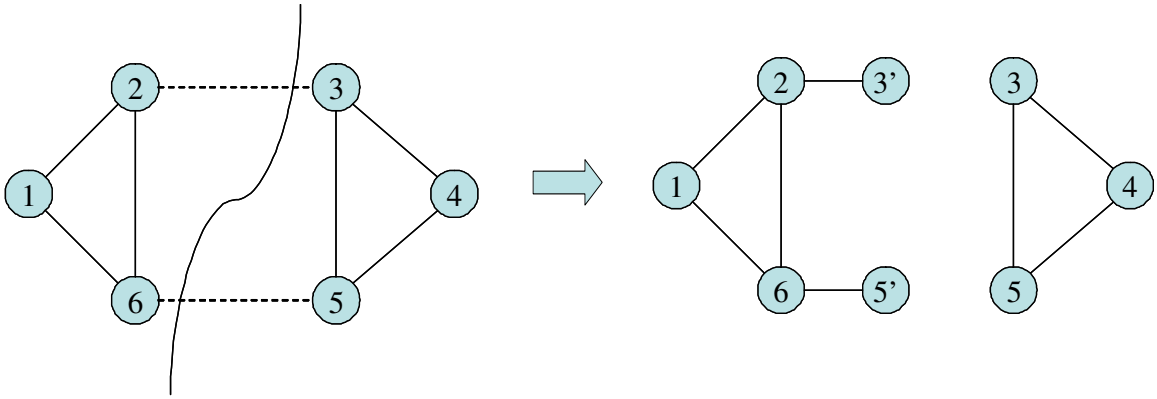


Figure 2: Vertex-Cloning: vertices 3 and 5 are cloned

contained within the feasible region of the vertex-disjoint formulation.

4.3.2 Bounds analysis: vertex-disjoint vs. edge-disjoint

In this section we investigate the relationship between arbitrary vertex and edge-disjoint partitions. We begin with an arbitrary edge-disjoint partition and an arbitrary vertex-disjoint partition with the same number of partitions. We employ a two - step process to transform the given vertex-disjoint partition into the given edge-disjoint partition. We show that the first step weakens the bound obtained from the vertex-disjoint partition and that the second step subsequently tightens this bound. However,

due to a lack of comparability between the first and second steps of the transformation, we cannot guarantee that this will result in a tightening of the bound. Figure 3 depicts arbitrary vertex and edge-disjoint partitions of the graph in Figure 1, each with two partitions. Figures 4 and 5 show Step 1 and Step 2, respectively, which transform the given vertex-disjoint partition into the given edge-disjoint partition.

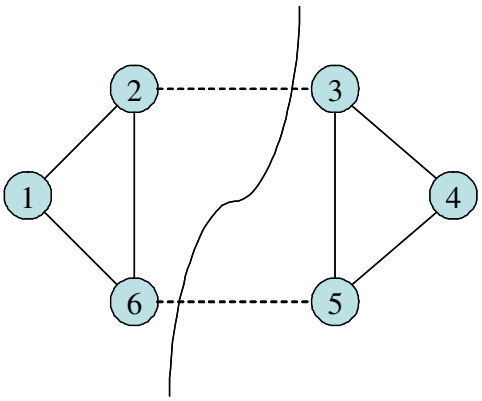


Figure 3a: An arbitrary vertex-disjoint partition

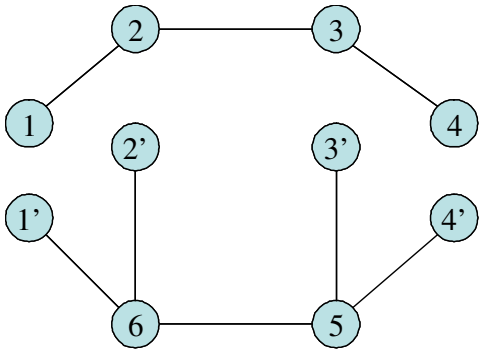


Figure 3b:An arbitrary edge-disjoint partition

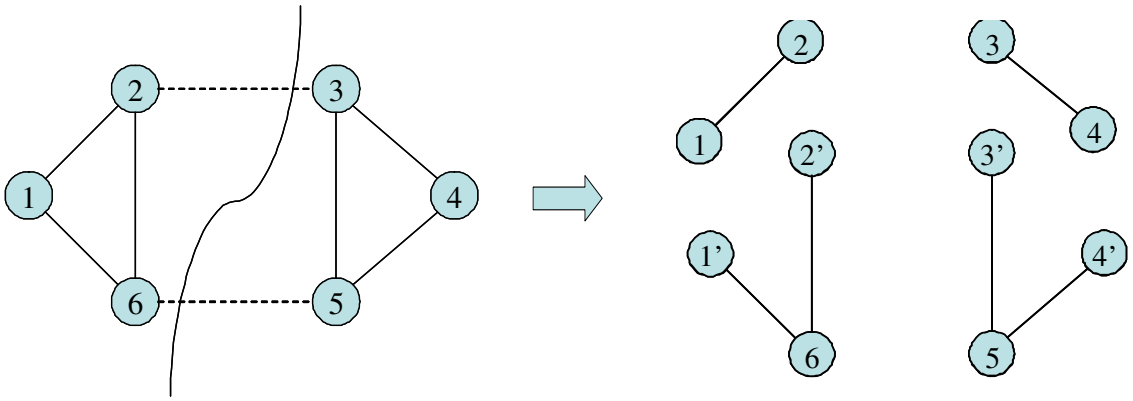


Figure 4: Step 1: Cloning vertices in the given vertex-disjoint partitioning to obtain an edge disjoint partitioning

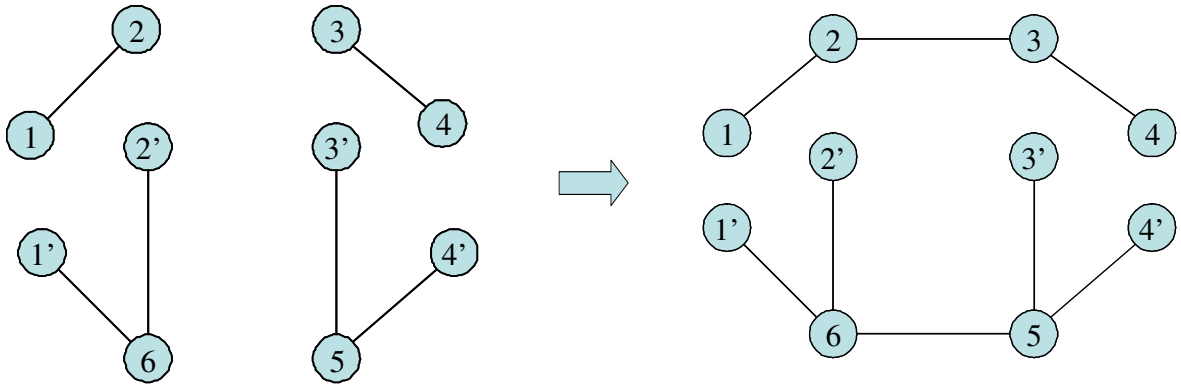


Figure 5: Step 2: Adding edges to transform the edge-disjoint partitioning of Figure 4 to obtain the given edge disjoint partition (Figure 3b)

We describe our analysis in further detail here. An arbitrary vertex-disjoint partitioning of a graph $G = (V, E)$ is defined by sets V_p for $p \in \{1,..,P\}$ and set $\hat{E}$ (refer to Chapter II). Likewise, an arbitrary edge-disjoint partitioning is defined by sets E_P for $p \in \{1,..,P\}$ and set $\tilde{V}$. The premise for this analysis is that the vertex - and edge-disjoint partitions are provided to us a priori. Our analysis is based on transforming the vertex-disjoint partition into the given edge-disjoint partition while monitoring the effect of the transformation on the associated bounds. Note that the bound obtained from solving the DWD reformulation associated with the vertex-disjoint partition is equal to the bound obtained from solving the fractional maximum weighted independent set problem (FMWISP) on the feasible region defined by $\left\{ L_{\hat{E}} \bigcap \left(\bigcap_p H_p^{VD} \right) \right\}$ [131]. We refer to this bound as the vertex-disjoint bound (VDB). For ease of notation we denote the intersection of the convex hulls of the vertex-disjoint subproblem polytopes ($\bigcap_p H_p^{VD}$)

by H_P^{VD}. This is equivalent to considering the vertex-disjoint subgraphs as one subgraph comprising disjoint components. From a theoretical point of view, this does not distort our analysis. Henceforth, we will refer to H_P^{VD} as the vertex-disjoint subproblem polytope. Thus VDB is equal to $\left\{L_{\hat{E}} \bigcap H_P^{VD}\right\}$. Similarly, the bound obtained from solving the DWD reformulation associated with the edge-disjoint partition is equal to the bound obtained from solving FMWISP on a feasible region defined by $\left\{L_{\tilde{V}} \bigcap \left(\bigcap_p H_P^{ED} \right)\right\}$ [131]. We refer to this bound as the edge-disjoint bound (EDB). Again we denote the intersection of the convex hull of the edge-disjoint subproblem polytopes ($\bigcap_p H_p^{ED}$) by H_P^{ED}. Thus EDB is equal to $\left\{L_{\tilde{V}} \bigcap H_P^{ED}\right\}$. Henceforth, we will refer to H_P^{ED} as the edge-disjoint subproblem polytope.

We begin by identifying the set $\tilde{V}$ corresponding to the given edge-disjoint partition ($\tilde{V} = \{1,2,3,4\}$ for the edge-disjoint partition depicted in Figure 3) and the corresponding set $p \in \tilde{P}_v$ for each $v \in \tilde{V}$ (in our example $\tilde{P}_1 = \tilde{P}_2 = \tilde{P}_3 = \tilde{P}_4 = \{1,2\}$). The first step (Figure 4) begins by using the above information to create $\left(\left|\tilde{P}_v\right| - 1\right)$ clones for each vertex $v \in \tilde{V}$ in the given vertex-disjoint subgraphs. The vertex-disjoint subproblem polytope H_P^{VD} is modified accordingly by introducing decision variables corresponding to clones and constraints equating decision variables corresponding to a vertex $v \in \tilde{V}$ and its clone. Introducing decision variables corresponding to clones increases the dimension of the subproblem polytope. This higher dimension subproblem

polytope is denoted H_P^{VD*}. However, adding equality constraints reduces the dimension of the subproblem polytope. We denote the convex hull of the feasible integer variables satisfying the equality constraints by $H_{\tilde{V}}$. The linear relaxation of $H_{\tilde{V}}$ is denoted $L_{\tilde{V}}$. The polytope corresponding to the subproblem with new decision variables corresponding to clones and the associated cloning equalities is thus denoted $H_P^{VD*} \bigcap H_{\tilde{V}}$. Note that invoking equality constraints within the subproblem ensures that H_P^{VD} is equal to $H_P^{VD*} \bigcap H_{\tilde{V}}$. Thus at this point, the bound obtained from DWD reformulation which is equivalent to the bound obtained by solving FMWISP on the feasible region defined by the set $\left\{L_{\hat{E}} \bigcap \left(H_P^{VD*} \bigcap H_{\tilde{V}}\right)\right\}$, is equal to VDB (since H_P^{VD} = $H_P^{VD*} \bigcap H_{\tilde{V}}$). To complete the first step, we relegate the equality constraints to the master problem. Accordingly the subproblem polytope now corresponds to H_P^{VD*}. At this point, the bound obtained from DWD reformulation corresponds to the bound obtained by solving FMWISP on the feasible region defined by the set $\left\{L_{\hat{E}} \bigcap L_{\tilde{V}} \bigcap H_P^{VD*}\right\}$. This bound is no stronger than VDB since we have replaced $H_{\tilde{V}}$ with its linear relaxation $L_{\tilde{V}}$. We refer to this bound as $\text{VDB}_{\text{STEP1}}$ ($\text{VDB}_{\text{STEP1}} \leq$ VDB).

In the second step (Figure 5), we relegate edges $\hat{E}$ to the subgraphs. Correspondingly edge-inequalities associated with $\hat{E}$ are introduced into to the subproblem. The subproblem polytope now corresponds to $\left\{H_{\bar{E}} \bigcap H_P^{VD*}\right\}$. The bound obtained from the DWD reformulation at the end of this step corresponds to the bound obtained by solving FMWISP on the feasible region defined by the set

$\left\{L_{\bar{V}} \bigcap \left(H_{\hat{E}} \bigcap H_P^{VD^*}\right)\right\}$ and is tighter than VDB_{STEP1} since we have replaced $L_{\hat{E}}$ with its integer convex hull $H_{\hat{E}}$. We refer to this bound as VDB_{STEP2}. Note that at the end of step 2 we obtain the edge-disjoint formulation associated with the given edge-disjoint partition (Figure 3b). Thus $VDB_{STEP2} = EDB$. We summarize the bounds associated with the two steps as follows:

$VDB \geq VDB_{STEP1}$

$VDB_{STEP2} \geq VDB_{STEP1}$

$EDB = VDB_{STEP2} \geq VDB_{STEP1}$

However, because $(VDB - VDB_{STEP1})$ may not be equal to $(EDB - VDB_{STEP1})$, it is not possible to specify the sign of (VDB - EDB) so that it is not possible to say that $EDB \leq VDB$ in all cases.

In the next section we provide a example, which shows that the bound obtained from an arbitrary edge-disjoint partition is weaker than that obtained from an arbitrary vertex-disjoint partition with the same number of partitions.

4.3.3 Example

The graph in Figure 1 provides the example which answers our initial question - 'does an arbitrary edge-disjoint partition yield a tighter bound than an arbitrary vertex-disjoint partition having the same number of partitions?'- in the negative. Solving the DWD reformulation on the vertex-disjoint partition depicted in Figure 3a we get an upper bound on the MWISP of 2. However, solving the DWD reformulation on the edge-disjoint partition depicted in Figure 3b, we obtain an upper bound of 3.

The optimal feasible bases for the associated DWD reformulations are as follows:

Vertex-disjoint:

- The optimal basis consists of columns associated with decision variables λ_1^1, λ_1^2, which both have the optimal value of 1. The respective cost coefficient for each of these columns is 1. Thus, the optimal solution ($Z^*_{RMP_{VD}}$) is equal to 2.
- The column associated with λ_1^1 corresponds to the independent set containing vertex {1} from subgraph 1 while that for λ_1^2 corresponds to the independent set containing vertex {3} from subgraph 2.

Edge-disjoint:

- The optimal basis consists of columns associated with decision variables λ_1^1, λ_2^1, λ_1^2, λ_2^2, each of which have the optimal value of 0.5. The cost coefficients for decision variables λ_1^1 and λ_2^1 are each 1, while the cost coefficients for decision variables λ_1^2, λ_2^2 are each 2. The optimal solution ($Z^*_{RMP_{ED}}$) is equal to 3.
- The columns associated with λ_1^1 and λ_2^1 correspond to independent sets containing vertices {1,3} and {2,4}, respectively, from subgraph 1, while those for λ_1^2 and λ_2^2 correspond to independent sets containing vertices {1',2',5} and {3',4',6}, respectively, from subgraph 2.

4.3.4 Example: vertex-disjoint vs. edge-cover

While in an edge-disjoint partitioning, an edge is present in only one subgraph in an edge-cover partitioning, an edge is replicated in every subgraph that contains both of

its end-points. In a later section we will show that edge-cover partitioning provides a tighter bound than the edge-disjoint partitioning. In this section, we answer the question -'does an arbitrary edge-cover partition yield a tighter bound than an arbitrary vertex-disjoint partition having the same number of partitions?'- in the negative by providing an example (Figure 6). The intuition behind the construction of this example is to obtain an optimal vertex-disjoint partitioning but a sub-optimal edge-cover partitioning.

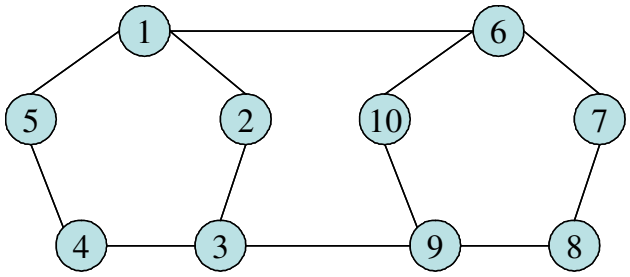


Figure 6: Construction for bounds analysis between Vertex-Disjoint and Edge-Cover

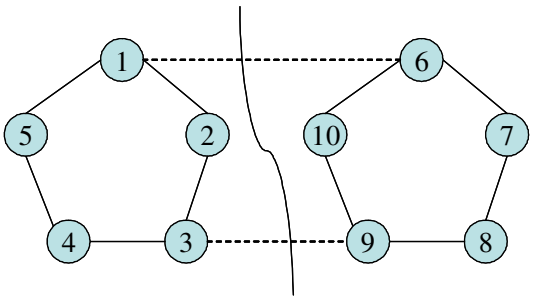


Figure 7a: Vertex-Disjoint Partitioning

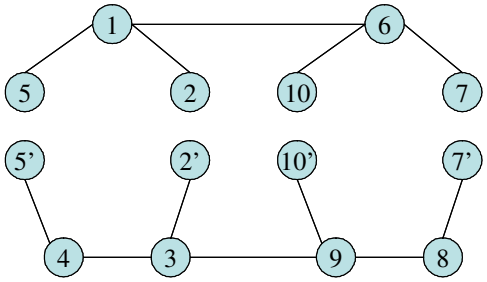


Figure 7b: Edge-Cover Partitioning

Note that the graph in Figure 6 is not facet producing and hence all facets of the corresponding MWISP polytope can be generated by lifting facets of polytopes

corresponding to subgraphs. We make leverage of this fact while partitioning the graph. The vertex-disjoint partitioning specified in Figure 7a provides an optimal bound of 4 upon solving the corresponding DWD reformulation whereas the edge-cover partitioning specified in Figure 7b provides a sub-optimal bound of 4.5. Note that the vertex-disjoint partitioning results in two subgraphs both of which are facet producing (odd-holes). Moreover, the subgraph induced by the corresponding cross-edges is a bipartite subgraph. This is the reason that the bound obtained is optimal. On the other hand, the edge-cover partitioning does not yield any special structure in its subgraphs. Moreover the subgraph induced the cloned vertices does not yield a clique. Hence, the edge-cover partitioning does not produce an optimal bound. Thus this example demonstrates that we can not guarantee that an arbitrary edge-cover partition will yield a tighter bound than an arbitrary vertex-disjoint partition having the same number of partition.

4.4. Edge-Disjoint partitioning

We use a tree decomposition approach [68] to partition the edge set $E(G)$ (suggested and implemented by a member of our research team). Given a graph G, the tree decomposition approach constructs a tree T that has $|E(G)|$ leaves. In addition, every non-leaf vertex in T has degree three. A bijective function υ maps the leaves of T to the edges of G. (T, υ) is referred to as the branch decomposition of G. Removing an edge e of T produces a vertex-disjoint partition of T into two subtrees. This consequently produces an edge-disjoint partition of G comprising two subgraphs - A_e and B_e, which

are induced by the edges incident to the leaves of the left and right trees, respectively. This is the basis for the edge-disjoint partitioning we employ.

We use two approaches for partitioning a graph – (p1) and (p2). (p1) is an edge-disjoint decomposition in which each edge appears in only one partition. In (p2) we replicate an edge in every subgraph that contains both of its end-points. This is referred to as the edge-cover approach since an edge can now be covered by more than one partition. (p2) offers two advantages. The first advantage is that the bound obtained from DWD reformulation associated with edge-cover decomposition is tighter than that obtained from edge-disjoint decomposition. This is because, in the edge-disjoint case since an edge appears in only one partition, it is possible that an edge (uv) is not invoked for a subgraph p which contains both u (or a clone of u) and v (or a clone of v). In such a case, a column entering RMP from subproblem p could incorrectly invoke both u and v (or their respective clones) as members of an independent set. By invoking an edge in all subgraphs that contain both its end-points, edge-cover decomposition eliminates this problem and guarantees that all columns entering RMP correspond to feasible independent sets of the original graph. Thus subproblem polytopes generated by edge-cover are contained within subproblem polytopes generated by edge-disjoint decomposition. The second advantage is that subproblem polytopes corresponding to edge-cover have fewer extreme points and consequently have fewer columns to be priced in comparison with edge-disjoint, thus reducing the computational effort.

4.5 RMP

RMP associated with DWD reformulation involves equality constraints and thus resembles a set-partitioning problem. Set-partitioning problems are much more challenging to solve than set-covering or set-packing problems. The size of RMP is critical to the computational effort required in solving an instance. Computational effort increases as the number of equalities increase. Thus, the size of $|\tilde{V}|$ is crucial for our approach. Moreover, set-partitioning constraints typically lead to a poor rate of convergence. We observe a similar behavior with our approach. We employ two alternative methods to deal with RMP. The first method (m1) simply invokes cloning equalities associated with $\tilde{V}$ in RMP. The primary advantage is its simplicity; however, it suffers computational disadvantages due to the resulting size of RMP and its degeneracy. The second method (m2) aims at identifying a minimal set of cliques of $G[\tilde{V}]$ that cover all vertices in $\tilde{V}$. We employ a best-in greedy heuristic to identify a set of cliques in $G[\tilde{V}]$. We aim to cover all vertices in $\tilde{V}$ by at least one clique. This method exploits a tighter formulation that results from incorporating clique inequalities, thus providing a better bound in comparision with (m1). It also reduces degeneracy in comparison with (m1).

4.6 Computational benchmarks

This section describes the results of our computational tests. We conduct our tests on unweighted instances from the Second DIMACS Implementation Challenge, (we actually use the complements of the listed graphs).

Table 7 compares (p1) and (p2) using (m1). The first five columns in Table 7 specify the instance: graph designation, the number of vertices, $|V|$; the number of edges, $|E|$; the % Density, Δ; and the number of partitions (P). The last five columns give the method [(p1) or (p2)] and the results of each: the total number of times RMP is solved (MP Sols); the number of B&B nodes required to find the optimal solution (B&B Nodes); the upper bound corresponding to the optimal solution of RMP at the root node (Z_{LP}); and the CPU run time for our B&P approach to prescribe an optimal solution (B&P Time). Results show that (p2) outperforms (p1). As discussed earlier (p2) offers two advantages. First, (p2) provides a better bound (see (Z_{LP})). Second, it has a better rate of convergence (see NMP). Based on this comparison, we select (p2) as the default for the remainder of our tests.

Table 8 compares performances of (m1) and (m2) using (p2). The first five columns in Table 8 specify the instance: graph designation, $|V|$, $|E|$, Δ, and P. The last six columns give the method [(m1) or (m2)] and the results of each: the number of cloning equalities, the number of clique inequalities, MP Sols, B&B Nodes, Z_{LP}, and B&P Time. (m2) is at a disadvantage because it must expend time to identify cliques. Moreover, for sparser graphs, no cliques exist in $G[\tilde{V}]$ and, hence, no improvement in

the bound is observed. For more dense instances, cliques are identified. For the three instances for which cliques were generated, there were marginal improvements in the computational effort in two (see Column 11), while there was no improvement in the remaining one (although the run-time using (m1) was low and did not provide much of an opportunity for improvement). Since the time required to generate cliques is not substantial and since, theoretically, there is sufficient advantage in generating cliques, we select (m2) as a default to use on other tests.

Based on this preliminary analysis, we henceforth use the (p2)-(m2) combination. Table 9 evaluates the sensitivity of our approach with respect to the number of partitions employed. The first four columns in Table 9 specify the instance: graph designation, $|V|$, $|E|$, and Δ. Columns 5-7 give the value of P employed, and the resulting $|\tilde{V}|$ and RMP rows, respectively. The last five columns presents relevant performance metrics: MP Sols, B&B Nodes, Z_{LP}, Z_{IP}, and B&P Time. Results are quite sensitive to the number of partitions. The computation effort is proportional to the number of equality constraints involved; hence, partitions resulting in fewer equality constraints tend to be more effective. Preliminary results also indicate that cloning fewer vertices tends to provide a tighter bound.

Table 10 evaluates our edge-disjoint approach further and compares it to the vertex-disjoint approach discussed in Chapter II. The first five columns in Table 10 specify the instance: graph designation, $|V|$, $|E|$, Δ, and P. The last six columns give the approach (edge-disjoint or vertex-disjoint) and the results of each: MP Sols, B&B

Nodes, Z_{LP}, Z_{IP}, and B&P Time. Results indicate that the vertex-disjoint approach is significantly better over most of the instances. The vertex-disjoint approach outperforms the edge-disjoint approach markedly on the denser instances (>30% density). This is primarily because, for denser instances, the rate of convergence for edge-disjoint DWD is extremely slow, due to the large number of equality constraints involved in RMP. A related difficulty for extremely dense graphs (>80%) is that the tree decomposition approach runs out of memory. For sparser graphs, our results indicate that the run time for the edge-disjoint approach is comparable to that required by the vertex-disjoint approach, primarily due to the fact that the former approach involves fewer equality constraints and thus the rate of convergence for DWD is better in application to sparser instances.

4.7 Conclusions

In this chapter we present an approach for solving MWISP by utilizing an edge-disjoint decomposition within a branch-and-price framework. We evaluate the effectiveness of this scheme computationally, providing insights into the advantages and disadvantages associated with it. We demonstrate that our approach is sensitive to parameters $\left|\tilde{V}\right|$ and $\left|\tilde{P}_v\right|$ $\forall v \in \tilde{V}$ as they govern the computational effort involved and the tightness of the bounds provided by the model. A small number of cloned vertices yields a tighter bound and is desirable. The computational effort is proportional to the number of equality constraints in RMP, each of which is associated with a cloned vertex. A larger number of equality constraints results in a slower rate of convergence for DWD.

Since denser graphs inevitably result in more equality constraints, our approach is more suitable for sparser graphs. The tree-decomposition partitioning scheme is not able to tackle large, dense graphs due to memory requirements. To be able to perform effectively over a wider spectrum of instances, we need to reduce the number of cloned vertices and substantially improve the rate of convergence of edge-disjoint DWD. In the next chapter we discuss convergence properties associated with DWD and adapt available techniques to improve the convergence performance of the edge-disjoint solver.

Table 7
Comparison of methods (p1) and (p2)

Instance	|V|	|E|	Δ	P	Method	MP Sols.	B&B Nodes	Z_{LP}	B&P Time (seconds)
hamming8-2	256	1024	3.1	20	(p1)	>612	**	**	**
					(p2)	495	1	128	4661.83
MANN_a9	45	72	7.3	5	(p1)	235	19	18.5	0.64
					(p2)	103	7	18	0.39
hamming6-2	64	192	9.5	8	(p1)	71	1	32	1.03
					(p2)	76	1	32	0.91
johnson8-4-4	70	560	23.2	3	(p1)	1109	12	16.21	40.08
					(p2)	164	1	14	8.4
johnson16-2-4	120	1680	23.5	8	(p1)	>23,945	>335	13.75	**
					(p2)	84	1	8	7.29
johnson8-2-4	28	168	44.4	8	(p1)	873	79	6.13	3.453
					(p2)	29	1	4	0.19

(p1) edge disjoint
(p2) edge cover
** exceeded run time limit of 1 hour

Table 8
Comparison of methods (m1) and (m2)

Instance	|V|	|E|	Δ	P	Method	# Equalities	# Clique rows	MP Sols.	Z_{LP}	B&P Time (seconds)
hamming8-2	256	1024	3.1	2	(m1)	78	0	3	128	0.23
					(m2)	78	0	3	128	0.23
MANN_a9	45	72	7.3	2	(m1)	8	0	15	17.5	0.09
					(m2)	8	0	15	17.5	0.09
hamming6-2	64	192	9.5	2	(m1)	21	0	3	32	0.03
					(m2)	21	0	3	32	0.03
Johnson8-4-4	70	560	23.2	2	(m1)	36	0	10	14	0.14
					(m2)	36	28	7	14	0.08
Johnson16-2-4	120	1680	23.5	2	(m1)	67	0	42	8	8.94
					(m2)	67	9	42	8	6.92
Johnson8-2-4	28	168	44.4	2	(m1)	17	0	16	4	0.03
					(m2)	17	5	19	4	0.03

(m1) no clique constaints
(m2) clique constraints

Table 9
Results for different number of partitions

Instance	|V|	|E|	Δ	P	$\lvert\tilde{V}\rvert$	RMP Rows	MP Sols.	B&B Nodes	Z_{LP}	Z_{IP}	B&P Time (seconds)
hamming8-2	256	1024	3.1	2	78	78	55	1	128	128	107.86
				20	197	557	495	1	128	128	4654.23
MANN_a9	45	72	7.3	2	8	8	256	21	17.5	16	0.89
				5	16	22	103	7	18	16	0.38
hamming6-2	64	192	9.5	2	21	21	6	1	32	32	0.34
				8	47	101	75	1	32	32	0.89
Johnson8-4-4	70	560	23.2	2	36	36	9	1	14	14	0.48
				3	51	66	145	1	14	14	6.83
Johnson16-2-4	120	1680	23.5	2	67	67	5	1	8	8	1.39
				8	111	375	88	1	8	8	6.94
Johnson8-2-4	28	168	44.4	2	17	17	5	1	4	4	0.08
				8	26	85	26	1	4	4	0.19

Table 10
Comparison of methods ED and VD on DIMACS instances

Instance	|V|	|E|	Δ	P	Approach	MP Sols.	B&B Nodes	Z_{LP}	Z_{IP}	B&P Time (seconds)
hamming8-2	256	1024	3.1	2	(ED)	55	1	128	128	107.86
				20	(VD)	8	1	128	128	0.44
MANN_a9	45	72	7.3	2	(ED)	256	21	17.5	16	0.89
				5	(VD)	187	22	18	16	0.41
hamming6-2	64	192	9.5	2	(ED)	6	1	32	32	0.34
				8	(VD)	4	1	32	32	0.06
Johnson8-4-4	70	560	23.2	2	(ED)	9	1	14	14	0.48
				3	(VD)	19	1	14	14	0.25
Johnson16-2-4	120	1680	23.5	2	(ED)	5	1	8	8	1.39
				8	(VD)	11	1	8	8	1.28
keller4	171	5100	35.1	2	(ED)	**	**	**	**	**
				5	(VD)	129613	14069	18.0 5	11	2367.05
hamming8-4	256	11776	36.1	2	(ED)	**	**	**	**	**
				8	(VD)	11	1	16	16	15.25
brock200-3	200	7852	39.5	2	(ED)	**	**	**	**	**
				2	(VD)	24734	2224	20	15	3451.84
Johnson8-2-4	28	168	44.4	2	(ED)	5	1	4	4	0.08
				8	(VD)	41	8	4	4	0.13
c-fat200-5	200	11427	57.4	2	(ED)	**	**	**	**	**
				7	(VD)	379	33	66.6 7	58	18.28
c-fat200-2	200	16665	83.7	2	(ED)	***	***	***	***	***
				4	(VD)	103	16	26.5	24	10.38
c-fat200-1	200	18366	92,3	2	ED	***	***	***	***	***
				2	VD	34	6	13	12	15.5

(ED) Edge-Disjoint Approach
(VD) Vertex-Disjoint Approach
** exceeded run time limit of 1 hour
*** exceeding memory limit

CHAPTER V

IMPROVING THE RATE OF CONVERGENCE OF DWD

5.1 Introduction

The slow rate of convergence associated with a DWD reformulation affects the time spent at each node in the B&P-tree and, hence, is critical to the efficiency of the approach. In this chapter, we develop insights into the convergence issues that accompany column generation. Specifically, we discuss the convergence properties of DWD and available techniques for improving the rate of convergence. We also present preliminary research towards developing a non-parametric approach to stabilizing DWD. Finally, we present techniques for improving the rate of convergence of the edge-disjoint B&P scheme discussed in Chapter IV. We present a computational evaluation of these specific techniques by conducting experiments on the linear relaxation associated with the root node of the edge-disjoint B&P scheme.

This chapter comprises six sections. Section 5.2 presents a brief overview of DWD and section 5.3 presents a dual perspective of DWD. Section 5.4 presents a brief discussion on DWD convergence issues. Section 5.5 discussed available techniques for improving the rate of convergence of DWD and presents insight into a non-parametric approach for stabilizing DWD. Section 5.6 presents specific techniques for improving the rate of convergence associated with the edge-disjoint B&P scheme and Section 5.7 presents a computational evaluation of these techniques.

5.2 DWD – overview

DWD entails decomposing the original problem into smaller subproblems and employing a master problem to coordinate the solutions proposed by these subproblems. The coordination is achieved in a price - directive fashion through the dual solutions provided by the master problem to the subproblems.

Below, we present an arbitrary linear formulation having a block diagonal structure, which is subsequently reformulated using DWD:

$$Z^* = \text{Max} \sum_{p \in P} w^p x^p \tag{5.1}$$

$$\sum_{p \in P} A_p x^p \leq b \tag{5.2}$$

$$D_p x^p \leq d^p \ \forall p \in P \tag{5.3}$$

$$x^p \in R^{n_p} \ , \forall p \in \{1, \ldots, |P|\}, \tag{5.4}$$

where A_p is the matrix of coefficients corresponding to x^p in inequalities associated with master problem constraints, D_p is matrix of coefficients corresponding to x^p in inequalities associated with subproblem constraints, $x^p \in R^{|n_p|}$ is the vector of decision variables associated with subproblem $p \in P$, $w^p \in R^{|n_p|}$ is the corresponding vector of weights, and n_p is the corresponding number of decision variables. This model is referred to as the original formulation.

The block-diagonal structure is exploited in the DWD reformulation (DWD) as follows:

$$Z^* = \text{Max} \sum_{p=1}^{P} \sum_{j \in J_p} \lambda_{jp} (w^p x^{jp}) \tag{5.5}$$

$$\sum_{p=1}^{P} \sum_{j \in J_p} \lambda_{jp} (A_p x^{jp}) \leq b \tag{5.6}$$

$$\sum_{j \in J_p} \lambda_{jp} = 1 \ \forall \ p \in \{1,..,P\} \tag{5.7}$$

$$\lambda_{jp} \geq 0 \ \forall \ p \in \{1,..,P\}, j \in J_p, \tag{5.8}$$

where J_p is the set of integer extreme points of subproblem polytope $Q_p = \left\{ x^p \in B^{|n_p|} : D_p x^p \leq d^p \right\}$, $x^{jp} \in B^{|V_p|}$ is the vector defining extreme point $j \in J_p$, and λ_{jp} is the decision variable corresponding to extreme point $j \in J_p$.

DWD (5.5-5.8) reformulation is solved using a column generation approach, which entails solving a restricted master problem (RMP) comprising a subset of columns. Pricing subproblems are solved to identify improving columns, which are entered into RMP in the subsequent simplex iteration. Pricing subproblems $p \in \{1,..,P\}$ are integer problems of the form:

$$Z_p^*(\alpha) = \text{Max} \left\{ \left(w^p - \alpha A_p^T \right) x^p : x^p \in Q_p \right\}, \tag{5.9}$$

where $\alpha \in R^{|\hat{E}|}$ is the vector of dual variables associated with the rows of constraint set (5.6) in RMP. A column corresponding to x^{jp} is deemed improving if $\left(w^p - \alpha A_p^T \right) x^{jp} - \beta_p > 0$, where β_p is the dual variable corresponding to the p^{th} convexity constraint (5.7).

At column generation iteration k the current solution obtained from RMP, denoted by Z_{RMP_k}, provides a lower bound (referred to as the *primal bound*). Note that, from strong duality, $Z_{RMP_k} = \alpha b + \sum_{p \in P} \beta_p$. Upon solving each of the pricing subproblems, an upper bound (referred to as the *dual bound*) is obtained: $Z_{RMP_k} + \sum_{p \in P} (Z_p^*(\alpha) - \beta_p)$ [17]. Thus, $Z_{RMP_k} \leq Z^* \leq Z_{RMP_k} + \sum_{p \in P} (Z_p^*(\alpha) - \beta_p)$. Optimality is guaranteed when no improving column is identified by any pricing subproblem; i.e. $\left(w^p - A_p^T \alpha\right) x^{jp} - \beta_p \leq 0 \ \forall \ p \in P$. Thus, the column generation scheme maintains primal feasibility and terminates upon achieving dual feasibility. While the primal bound is guaranteed to improve monotonically (except for degenerate iterations), the dual bounds are not guaranteed to improve monotonically. In fact, the erratic nature of the dual bound is a concern. To lend further insight into the principles of DWD, we describe the dual perspective in the next section.

5.3 DWD: the dual perspective

The dual of RMP can be formulated as follows:

$$\min_{u \geq 0, v_0, w_0} \ b\ \alpha + \sum_{p \in P} \beta_p \tag{5.10}$$

s.t.

$$w^p x^{jp} - \alpha \left(A_p x^{jp}\right) \geq \beta_p \qquad \forall \ j \in J_p, \ p \in P \tag{5.11}$$

This can alternatively, be written as [43]

$$\min_{u \geq 0} \; b\ \alpha + \sum_{p \in P} \Theta^p(\alpha) \tag{5.12}$$

where $\Theta^p(\alpha) = \left\{ w^p x^{jp} - \alpha\left(A_p x^{jp}\right) \right\} \}\ \forall\ p \in P$. (5.13)

Note that (5.12) has the form of a non-differentiable optimization problem. Specifically, dual problem (5.12)-(5.13) represents the minimization of a convex, piecewise-linear function. At a differentiable point, the epigraph of this function has a unique supporting hyperplane and the corresponding slope of this hyperplane is the gradient [108]. At a non-differentiable point (a point at which two or more of the piecewise functions intersect), the epigraph has an infinite set of supporting hyperplanes [108]. The slope of a supporting hyperplane is referred to as the *sub-gradient* and the set of all such sub-gradients is called the *sub-differential* [108].

Cutting plane algorithms are widely used for solving non-differentiable optimization problems. Such an algorithm uses an oracle to dynamically generate supporting hyperplanes to approximate the epigraph of the non-differentiable function. From a dual perspective, DWD column generation represents such a cutting plane algorithm, specifically Kelley's cutting plane algorithm [78]. The optimal dual solution obtained at each iteration of DWD column generation corresponds to the minimum of the current (i.e., employing the cutting planes generated so far) piecewise approximation of the dual function (5.12). Columns in the primal correspond to hyperplanes that support the epigraph of the dual function (5.12). Thus, from the dual perspective, at every iteration a new supporting hyperplane is generated to cut off the current optimal dual solution. For the dual function (4.12), the sub-gradient of the supporting hyperplane

is given by $\left\{b - \sum_{p \in P} \left(A_p x^{jp}\right)\right\}$. Optimality is guaranteed when no cut can be generated at the current dual solution, implying that the minimum of function (5.12) has been attained. This is exactly Kelley's cutting plane algorithm.

This dual point of view provides significant insights into the functioning of DWD. The convergence issues observed with DWD are similar to the convergence issues observed with Kelley's cutting plane algorithm; in fact, both use the current optimal dual solution to generate an improving column. Using the current optimal dual solution is critical in defining the convergence of the algorithm. The primary concern is that, if the primal RMP is degenerate, the dual has alternative optimal solutions. DWD employs extreme point dual solutions to generate columns for RMP but they may not facilitate convergence. Previous research has shown that in the case of primal degeneracy, an inner point with respect to the optimal dual face could be more suitable for generating improving columns. Such variants of cutting plane algorithms that use alternative dual points are prevalent for non-differentiable optimization problems (e.g., the Analytic Center Cutting Plane method (ACCPM) chooses the analytic center of the current approximation of the epigraph to generate improving columns). In the next section, we describe the primary issues affecting the rate of convergence of DWD.

5.4 DWD: convergence issues

Typically, DWD converges slowly. Four main phases that affect the rate of convergence are defined as follows:

Heading-In relates to the number of iterations required to identify an initial feasible basis and is a pronounced issue for set partitioning problems for which column generation spends a substantial amount of time to identify an initial feasible basis.

Oscillation occurs when dual solutions obtained from solving RMP oscillate with no well-defined pattern. This results in erratic changes of the corresponding dual bounds. The optimal dual solution obtained from RMP corresponds to the minimum of the current approximation of the dual function, which is refined by including the cutting plane generated by this dual solution. Oscillation occurs because the new minimum of the refined dual function is not guaranteed to be close to the previous minimum. It appears that DWD would converge more rapidly if successive dual solutions progressed smoothly to an optimal solution. This would also provide successive dual bounds that improve monotonically.

Primal degeneracy results when an improving column enters the RMP basis but does improve the primal bound. A degenerate primal solution corresponds to alternative optimal dual solutions in the dual space. A column newly entered in the primal model corresponds to a new cut in the dual space. This cut renders the current extreme point dual solution infeasible but does not necessarily cut off all alternative dual optima. Consequently, the primal bound does not improve on a degenerate iteration.

Tailing-off effect is the phenomenon that occurs as DWD approaches an optimal solution; it requires a substantial amount of time to close the gap between primal and dual bounds. The slow rate of convergence due to tailing-off is notably severe for set partitioning problems.

5.5 Techniques for improving convergence of DWD

In this section we discuss prevalent techniques for accelerating the rate of convergence of DWD and present a theoretical insight into a non-parametric method for stabilizing DWD.

5.5.1 Initializing RMP

Artificial variables are used in the initial RMP basis for the case in which a master problem incorporates equality constraints. Since the value "Big M", which is assigned to the objective function coefficient associated with each artificial variable corresponds to a bound on the associated dual variable, a tight estimate of Big M often aids in the rapid convergence to a feasible (primal) solution that does not include artificial variables. The rate of convergence can also be accelerated if an initial set of columns can be generated to provide a good approximation of the epigraph of function (5.12) near its minima. A good heuristic solution to the integer problem does not necessarily provide a good estimate of the optimal DWD primal bound; hence, adding columns prescribed by a heuristic do not necessarily improve the rate of convergence.

5.5.2 Stabilizing DWD

Stabilization seeks to avoid erratic oscillation of dual variable values. The main idea behind stabilization is to restrict each dual variable to take values within a specified trust-region. The optimization of the dual function is restricted within this trust-region, which is redefined appropriately as the algorithm converges. Smoothing approaches have also been proposed to capture the history of the column generation process by

using some combination of previously generated dual solutions along with the current dual optimal solution. In this section we describe these techniques in more detail.

Boxstep method: In the Boxstep method [90], optimization in the dual space is explicitly restricted to a box obtained by enforcing upper and lower bounds on dual variable values. Solving a series of optimization problems, each around a more refined box, stabilizes the column generation process. If the optimal dual solution associated with a particular box does not lie completely within it, the center of the box is updated to the current optimal dual solution and the revised problem is optimized. If the current optimal dual solution lies completely within the box, we have attained global optimality and primal feasibility is guaranteed. Our preliminary analysis of the Boxstep method reveals that the box width is critical - a box-size that is too small may require more frequent updates, while one that is too large may not improve convergence. However, the best box-size is relative to the instance at hand. Preliminary analysis also show that different box sizes, each containing the optimal dual solution, need not lead to similar convergence rates. Ideally, the smallest box containing the optimal dual is the desired option because it provides the least opportunity for oscillation and will thus require just one problem to be optimized (i.e., one box). The important concerns affecting the efficiency of the Boxstep method are to provide a good starting dual solution (which is close to the optimal dual solution) and to prescribe good box-sizes a priori. A successful implementation of Boxstep requires effective resolution of these concerns. In the last section we present our adaptation of Boxstep, addressing these concerns relative to the edge-disjoint scheme.

3-Piece: In the Boxstep method, dual variable values are not permitted to violate the trust-region; they must lie within the box. However, in a related approach [43], a linear penalty function comprising three pieces is invoked and dual variables are allowed to a take values outside the box, but at the expense of incurring a penalty. Similar to the Boxstep method, deciding the parameters of the penalty function defines the efficiency of this approach.

Wentges smoothing approach: The main idea of the smoothing approach is to maintain proximity to the best dual solution obtained so far (the one corresponding to the best dual bound). In the Wentges approach [129], the next dual vector used to generate improving columns is obtained by taking a step away from the current dual solution in the direction of the best dual solution. As the algorithm converges, it places an increasing emphasis on the best dual solution found.

5.5.3 A new non-parameteric approach for stabilizing

All previously developed techniques involve parameters for which it is difficult to determine effective values. In this section we explore a non-parametric approach for stabilizing DWD and discuss some concerns related to it. Our approach is based on insights from interior point approaches that have been used to stabilize DWD convergence. Most interior point algorithms (e.g analytical center, volumetric center, etc.) use a central point with respect to the current approximation of the epigraph of the dual objective function. A central point is deemed useful because it summarizes dual

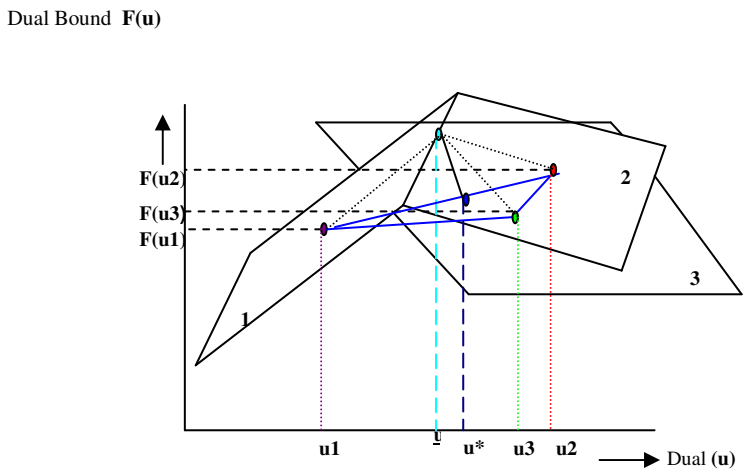


Figure 8: Non-Parametric approach to stabilization

information accumulated during the progress of the algorithm. Therefore, calculating a central point, which is not easy, is the focus of most research in this area. In addition, interior point algorithms do not apply to DWD, which uses the Simplex method. In this section we explore an alternative method that employs the idea of a central point within the framework of the Simplex method. Our approach uses a projection of the current optimal dual solution provided by RMP (Figure 8). This optimal dual solution corresponds to an intersection of hyperplanes in the dual space. We refer to a hyperplane that intersects at the current optimal dual solution as an active hyperplane. Each hyperplane supports the dual epigraph at a point corresponding to the dual solution which was used in generating this hyperplane. We refer to this point corresponding to an active hyperplane as an active dual point. We construct the affine hull of all the active dual points and project the optimal dual solution obtained from RMP onto it. The motivation for this approach is that the dual solution corresponding to this projected

point has a central property because it captures information provided by the set of active dual points obtained in previous iterations. More importantly, the projected point tends to be close to the better dual values ("better" in the sense of the dual bounds associated with each). The optimal dual solution provided by RMP is indifferent to the quality of previous dual solutions but is dependent (only) on the slopes of supporting hyperplanes generated by these dual solutions. In using the slope of the affine hull, our goal is to implicitly force dual solutions to be close to one another on subsequent iterations, thus preventing excessive oscillation. Finally, the projected dual point can be obtained within a Simplex-method framework as shown below.

Calculating the projected dual: The projection of point $\hat{x}$ on to the affine hull of k points x_i, $i = 1,..k$ is obtained by solving the following minimization problem:

$$\min \{ \| \sum_{i=1..k} \lambda_i x_i - \hat{x} \| \}^2) \tag{4.14}$$

s.t.

$$\sum_{i=1..k} \lambda_i = 1. \tag{4.15}$$

Model (4.14)-(4.15) can be solved using the Karush-Kuhn-Tucker conditions [17]. The corresponding lagrange function can be formulated as follows:

$$L(\lambda, \mu) = \sum_j (\sum_{i=1..k} (\lambda_i x_{ij}) - \hat{x}_j)^2 + \mu (1 - \sum_{i=1..k} \lambda_i),$$

where $\hat{x}_j$ is the jth component of vector $\hat{x}$; and x_{ij} is the jth component of vector x_i. The associated KKT conditions are

$$\frac{\partial}{\partial \lambda_i} (L(\lambda, \mu)) = \sum_j 2*(\sum_{i=1..k} (\lambda_i x_{ij}) - \hat{x}_j) x_{ij} - \mu = 0 \text{ for } i = 1, \ldots, k$$

and

$$\frac{\partial}{\partial \mu}\left(L(\lambda, \mu) \right) = 1 - \sum_{i=1..k} \lambda_i = 0.$$

Solving these (k + 1) equations in (k+1) unknowns (μ and λ_i for $i = 1, \ldots, k$) we obtain the projected dual.

Our primary concern is that, often, we use columns generated a priori (e.g., by a primal heuristic) to initiatize RMP. The dual points that correspond to such columns are not known. Thus, when hyperplanes corresponding to these columns are active in defining the current optimal dual, the corresponding affine hull is not well-defined because it does not include any dual points corresponding to these columns. Our future research will try to overcome these difficulties and explore non-parametric approaches further.

5.6 Improving convergence of edge-disjoint DWD

In this section we describe specific techniques we implemented to improve the convergence of the DWD reformulation of the edge-disjoint partitioning scheme (see Chapter IV).

5.6.1 An improving initial set of columns

We draw insight for generating a set of initial columns from the unique structure of the master problem constraints for the edge-disjoint scheme. Each equality constraint involves two decision variables: one related to a cloned vertex; and the other, to its clone. The coefficient of the former is +1 while that of the latter is -1. This implies that

every hyperplane that supports the epigraph of the dual function will have values +1, 0, or -1 as components of its gradient. The component of the gradient corresponding to an equality constraint will take a value of +1 if the corresponding generated columns involve the cloned vertex but not the clone. The component takes a value of -1 if the corresponding generated columns do not involve the cloned vertex but does the clone. Finally the gradient has a component value of 0 if the corresponding generated columns involve both a cloned vertex and its clone.

We use this insight to generate useful columns apriori. A hyperplane that has a gradient component of +1 associated with an equality constraint can be obtained by generating a column that involves the cloned vertex. Similarly, by generating a column that involves a clone, we can generate a hyperplane with a gradient component of -1, which is associated with the equality constraint. We initiate RMP with columns based on this criterion. Preliminary tests show that such initial columns indeed accelerate the convergence of DWD.

5.6.2 Defining optimal bounds on the dual variables

An ideal implementation of the boxstep method would utilize the optimal box-width. In this section we exploit an observation regarding the edge-disjoint scheme to create tight dual bounds. Our scheme is based on the insight obtained from observing the structure of master problem constraints. We illustrate our insight using an example. Assuming a vertex v has 2 clones v' and v'', the associated equality constraints are as follows: $x_v - x_{v'} = 0$ and $x_v - x_{v''} = 0$**.** The cost coefficients of the decision variables x_v**,**

$x_{v'}$, $x_{v''}$ are $\frac{w_v}{3}$ each, where w_v is the original weight associated with vertex v. Let δ' and δ'' denote the dual variables associated with the cloning equalities. Employing duality we have the following constraints on the dual variables: $\delta' + \delta'' \leq \frac{w_v}{3}$, $-\delta' \leq \frac{w_v}{3}$, and $-\delta'' \leq \frac{w_v}{3}$. These constraints imply that dual variables δ' and δ'' are bounded within the interval [$-\frac{w_v}{3}$, $2\frac{w_v}{3}$]. In general, for a cloned vertex with $\left|\tilde{P}_v\right| - 1$ clones, the dual variables associated with the corresponding equality constraints are bounded within the interval [$-\frac{w_v}{3}$, $(\left|\tilde{P}_v\right| - 1) * \frac{w_v}{\left|\tilde{P}_v\right|}$]. We use this observation to enforce bounds on the dual variables within the Boxstep method. Preliminary tests show that these dual bounds promote convergence.

5.6.3 A relaxation scheme

For denser instances, the tailing-off effect is severe and convergence typically takes a number of hours at the root node for the edge-disjoint approach. The last method aims at solving a relaxation of RMP in order to improve performance. Standard techniques for relaxing constraints of the set-partitioning type involve penalizing a violation of the equality constraint by using surplus and slack variables or by perturbing the equations. We explore a related relaxation for the edge-disjoint scheme, which offers the potential of allowing better control and involving fewer parameters.

The equality constraints that relate each vertex $v \in \hat{V}$ and its clones (see Chapter III) contribute substantially to the computational effort required to solve the edge-disjoint DWD formulation:

$$S_{\tilde{v}} = \left\{ x \in B^{|V \setminus \tilde{V}| + |\tilde{V}| \times \sum_{v} |P_v|} : x_{v_k} - x_{v_{1'}} = 0, \forall k \in \{2,..,\left|\tilde{P}_v\right|\}, v \in \tilde{V} \right\}. \quad (4.5)$$

Thus, RMP can be decomposed into disjoint sets of equality constraints, each set involving a cloned vertex $v \in \hat{V}$ and its clones. We focus on each such set individually. For each $v \in \hat{V}$, we have $\left|\tilde{P}_v\right| - 1$ equality constraints; for example, a vertex v having two clones v' and v'' is associated with two equality constraints: $x_v - x_{v'} = 0$ and $x_v - x_{v''} = 0$. The solution space associated with these equality constraints corresponds to the diagonal of the unit hypercube formed by the binary variable x_v and its clones, $x_{v'}$ and $x_{v''}$ (Figure 9a). Since such a feasible region is highly restrictive, extensive computational effort is entailed. Based on this observation, we relax the solution space to a box of width δ around this diagonal (Figure 9b). This relaxation has the advantage that it is both intuitive and simple to implement. Moreover, the relaxation is easy to control – we can enlarge or reduce the boxsize, depending on how challenging an instance is.

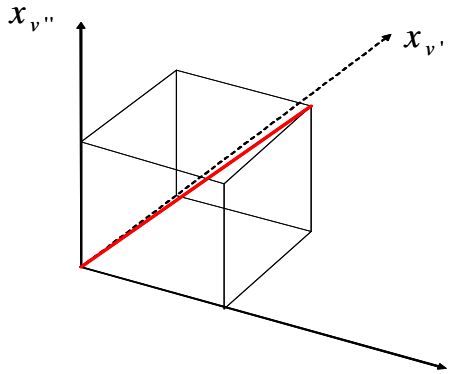


Figure 9a: The solution space corresponding to the equality constraints involving a vertex and its clones

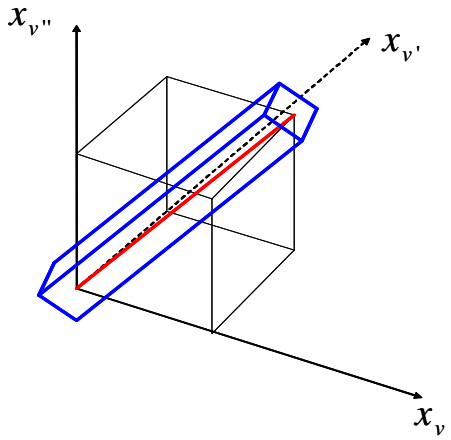


Figure 9b: Relaxing the solution space

5.7 Computational benchmarks

In this section we present a computational evaluation of our stabilization methods in application to the edge-disjoint B&P approach for MWISP (see Chapter IV), focusing on instances from the Second DIMACS Implementation Challenge (we actually use the complements of the listed graphs). We present results based on applying our stabilization methods at the RMP root node. Table 11 presents results; column 1 identifies the graph involved in each test and columns 2-7 describe the instance, giving, respectively, the number of vertices $|V|$; the number of edges, $|E|$; the % Density, Δ; the number of partitions (P), the corresponding $|\tilde{V}|$; and the number of equality constraints in RMP (Equalities). Column 8 identifies the stabilization method. Columns 9-11 present relevant performance metrics with respect to the rate of convergence of DWD: the root

node solution (Z_{LP}), the total number times RMP is solved (MP Sols), and the CPU run time for our B&P approach to prescribe the root node RMP optimal solution (Time).

Table 11 compares the performances of the techniques described in section 5.6.1, 5.6.2 and 5.6.3 in comparison with the unstabilized version. For each instance, five rows present results obtained from applying (m1): the unstabilized DWD, (m2): the initial set of columns as discussed in section 5.6.1, (m3): the dual bounds within a Boxstep method as described in section 5.6.2 in addition to employing the initial set of columns, and the relaxation described in section 5.6.3 using box-widths of 0.1 (m4) and 0.05 (m5). Note that (m4)-(m5) relax the problem and hence the upper bounds obtained from the linear relaxation (Z_{LP}) are weaker than that obtained from (m1)-(m3).

Results indicate that the time expended in generating the initial set of columns for (m2) is more than offset by the improvement in the rate of convergence. By employing the dual bounds within a Boxstep method, (m3) improves the rate of convergence of DWD. For instance keller4 (a large, dense instance), inspite of employing (m2) and (m3) techniques, convergence was not attained at the root node within a realistic amount of time. The main reason for this poor performance was the tailing-off effect, which was addressed by the relaxation strategy using (m4) and (m5). However, the relaxation resulted in a weaker upper bound (see Z_{LP}). For sparser instances, the upper bounds obtained by (m4) and (m5) are tight in comparison with that obtained by (m1)-(m3), but the bound weakens as density increases. With respect to run time, (m4) and (m5) together provide the best results for 5 of the 8 instances and the worst only on instance johnson16-2-4. This is intuitive since (m4) and (m5) solve a

relaxation of the original problem. Instances MANN_a9, hamming6-2, and johnson8-2-4 do not provide much opportunity to make improvements with respect to run time.

5.8 Conclusions

In this chapter we describe available techniques employed to accelerate convergence of DWD. We present preliminary research to explore a new non-parametric approach for stabilizing DWD. Finally, we demonstrated adaptations of available techniques in application to the edge-disjoint B&P formulation. Our results indicate that our implementations were successful in improving the rate of convergence. Our future research will explore more generic strategies for set-partitioning problems and explore efficient techniques for improving the convergence of the edge-disjoint B&P formulation.

Table 11
Comparison of methods (m1), (m2), (m3), (m4) and (m5)

Instance	|V|	|E|	Δ	P	$\lvert\tilde{V}\rvert$	Equalities	Method	Z_{LP}	MP sols.	Time (seconds)
MANN_a27	378	702	0.01	25	70	284	(m1)	135	101	5.88
							(m2)	135	61	2.68
							(m3)	135	84	9.34
							(m4)	137.4	29	3.17
							(m5)	136.25	28	2.77
hamming8-2	256	1024	3.1	20	197	557	(m1)	128	495	4547.45
							(m2)	128	403	3442.38
							(m3)	128	292	1741.8
							(m4)	128.925	21	4.67
							(m5)	128.47	35	6.86
MANN_a9	45	72	7.3	5	16	22	(m1)	18	21	0.06
							(m2)	18	12	0.06
							(m3)	18	10	0.05
							(m4)	18.37	7	0.03
							(m5)	18.18	8	0.05
hamming6-2	64	192	9.5	8	47	101	(m1)	32	75	0.58
							(m2)	32	67	0.86
							(m3)	32	4	0.05
							(m4)	32.33	3	0.06
							(m5)	32.12	3	0.04
johnson8-4-4	70	560	23.2	3	51	66	(m1)	14	145	6.57
							(m2)	14	142	6.74
							(m3)	14	172	9.15
							(m4)	14.43	4	0.39
							(m5)	14.22	5	0.47
johnson16-2-4	120	1680	23.5	8	111	375	(m1)	8	88	5.36
							(m2)	8	56	4.96
							(m3)	8	46	5.37
							(m4)	12.784	35	37.57
							(m5)	11.58	75	12.89
keller4	171	5100	35.1	8	157	621	(m1)	13.63	>500	**
							(m2)	13.63	>500	**
							(m3)	13.63	373	4446.52
							(m4)	16.79	55	648.59
							(m5)	15.95	101	757.43
johnson8-2-4	28	168	44.4	8	26	85	(m1)	4	26	0.11
							(m2)	4	14	0.05
							(m3)	4	11	0.06
							(m4)	5.53	10	0.09
							(m5)	4.938	13	0.09

(m1) unstabilized
(m2) employing starting set of columns
(m3) employing dual bounds with Boxstep
(m4) 0.1 relaxation
(m5) 0.05 relaxation
** exceeds run-time limit of 3 hours

CHAPTER VI

CUT GENERATION WITHIN B&P – A LIFTING TECHNIQUE

6.1 Introduction

DWD reformulation can provide a tighter bound than that given by the LP relaxation of a model. However, in a typical implementation of B&P, the bound obtained may not tight enough to solve challenging instances effectively. B&P can potentially be enhanced by incorporating cutting planes to form a branch-and-cut-and- price (BCP) approach. However, incorporating cutting planes within B&P is challenging. In this chapter we focus on generating valid linear inequalities that can be incorporated in RMP to tighten the formulation. Note that traditional techniques for deriving generic cutting planes from the optimal Simplex tableau – Gomory and L&P cutting planes – will, within a DWD framework, generate cutting planes in terms of the RMP decision variables. A cutting plane in terms of RMP decision variables entails the disadvantage that it can distort subproblem structure. Hence, the challenge is to present techniques for generating cutting planes in terms of the original problem variables. This is the precise reason why cutting plane techniques are not used routinely in the B&P framework.

In this chapter we introduce a generic lifting technique for deriving cutting planes in the B&P framework in terms of the original problem variables. Moreover, our technique does not rely on the polyhedral properties of the underlying problem. Although we discuss a specific application to MWISP, our approach is useful in generic applications of B&P. We begin our approach by identifying faces/facets of the subproblem that are tight at the current DWD solution using a modification of the facet generation procedure (FGP) [102]. These valid inequalities, however, are of no use if

incorporated in RMP because they have been implicitly invoked; DWD reformulation optimizes over the integer convex hull of each subproblem. However, we show that these valid inequalities - when lifted over variables associated with other subproblems- can potentially generate valid inequalities that cut off the current fractional solution. Within the context of MWISP, this method corresponds to identifying facets of the polytopes associated with G that are obtained by lifting facets of polytopes associated with subgraphs of G. This is the basis of our cut generation scheme.

This chapter has five sections. Section 6.2 presents relevant formulations for our method; and section 6.3 discusses the lifting scheme. Sections 6.4 and 6.5 present computational results and conclusions, respectively.

6.2 B&P formulations

In this section we present a generic formulation having a block diagonal structure, which is amenable to DWD. The corresponding feasible region is thus represented by the following set of constraints:

$$\begin{aligned} & A_1 x^1 + A_2 x^2 + \cdots + A_{|P|} x^{|P|} \le b \\ & D_p x^p \le d^p \qquad \forall p \in P \\ & x^p \in B^{n_p} \qquad \forall p \in P, \end{aligned} \tag{6.1}$$

where A_p is the matrix of coefficients corresponding to x^p in master problem constraints, D_p is the matrix of coefficients corresponding to x^p in inequalities associated with subproblem p, $x^p \in B^{n_p}$ is the vector of decision variables associated

with the n_p variables in partition $p \in P$, and $w^p \in R^{n_p}$ is the corresponding vector of weights.

The block-diagonal structure of formulation (6.1) is exploited in the following DWD reformulation (DWD):

$$\sum_{p=1}^{P} \sum_{j \in J_p} \lambda_{jp} (A_p x^{jp}) \leq b \tag{6.2}$$

$$\sum_{j \in J_p} \lambda_{jp} = 1 \qquad \forall\ p \in \{1,..,P\} \tag{6.3}$$

$$\lambda_{jp} \geq 0 \qquad \forall\ p \in \{1,..,P\},\ j \in J_p, \tag{6.4}$$

where J_p is the set of integer extreme points of $Q_p = \left\{x^p \in B^{|n_p|} : D_p x^p \leq d^p\right\}$, $x^{jp} \in B^{|V_p|}$ is the vector defining extreme point $j \in J_p$, and λ_{jp} is the RMP decision variable corresponding to extreme point $j \in J_p$. Subproblem $p \in \{1,..,P\}$ is an integer problem of the form:

$$Z_p^*(\alpha) = \text{Max}\left\{\left(w^p - A_p^T \alpha\right) x^p : x^p \in Q_p\right\}, \tag{6.5}$$

in which $\alpha \in R^{|\hat{E}|}$ is the vector of dual variables associated with the rows of constraint set (6.2). A column corresponding to x^{jp} is deemed improving if $\left(w^p - A_p^T \alpha\right) x^{jp} - \beta_p > 0$, where β_p is the dual variable corresponding to the p^{th} convexity constraint (6.3).

This DWD reformulation involves λ_{jp} decision variables, which differ from those in original formulation (6.1). Consequently, the optimal Simplex tableau corresponding to the DWD reformulation is in terms of λ_{jp} rather than x^p. Traditional cutting plane techniques - Gomory and L&P cutting planes – exploit the optimal Simplex tableau.

Relative to the DWD reformulation, this results in cutting planes involving λ_{jp} decision variables. The disadvantage of such a scheme is that resulting cutting planes might change the subproblem structure, posing a challenge to the subproblem solver and affecting the overall performance of B&P. To improve the B&P approach, the challenge is to present techniques for generating cutting planes in terms of the original problem variables. Invoking such cutting planes in RMP will not distort subproblem structure.

We next describe a formulation that provides insight into our lifting technique. The DWD reformulation implicitly invokes the integer convex hull of each subproblem polytope. Assuming that we have a minimal representation of the integer convex hull of each subproblem polytope, an equivalent representation (DWD") is as follows:

$$A_1 x^1 + A_2 x^2 + \cdots + A_{|P|} x^{|P|} \le b \tag{6.6}$$

$$\tilde{D}_p x^p \le \tilde{d}^p \qquad \forall p \in P \tag{6.7}$$

$$x^p \in R^{n_p} \qquad \forall p \in \{1,\ldots,|P|\}, \tag{6.8}$$

where (6.7) is the minimal representation of $Q_p = \left\{x^p \in B^{|n_p|} : D_p x^p \le d^p\right\}$. Formulation (6.6) – (6.8) provides the same bound as formulation (6.2) – (6.4). Moreover, the optimal feasible bases of these formulations correspond to each other. The difference between the two formulations is that, while DWD reformulation invokes the convex hulls of integer subproblem polytopes implicitly, DWD" invokes them explicitly in (6.7). In the next section we describe our lifting technique, which generates cutting planes in terms of the original variables.

6.3 Lifting technique

Let $\bar{x}$ ($[\bar{x}^1, \bar{x}^2, \ldots, \bar{x}^{|P|}]^T$) denote the current DWD fractional solution obtained by the transformation $\bar{x}^p = \sum_{j \in J_p} \bar{\lambda}_{jp} x^{jp}$, in which $\bar{\lambda}$ is the vector of the optimal basic variables in RMP. Since $\bar{x}$ corresponds to an optimal feasible basis for DWD", a subset of constraints (6.6) and (6.7) are tight at $\bar{x}$. Note that, while it is easy to determine the subset of constraints (6.6) that are tight at $\bar{x}$, we can not do the same for constraints (6.7) because we do not know them explicitly. Further, we reiterate that constraint set p in (6.7) includes decision variables associated with only one partition. The following theorem presents the basis for our lifting technique.

Theorem 6.1: Suppose that inequality $\alpha^{p_i} x^{p_i} \leq \beta$ from constraint set (6.7) is tight at $\bar{x}$ for some partition $p_i \in P$. Further, suppose that lifting $\alpha^{p_i} x^{p_i} \leq \beta$ over fractional variable x_k (i.e., $0 < \bar{x}_k < 1$) generates the valid inequality $\alpha^{p_i} x^{p_i} + \alpha_k x_k \leq \beta$ with $\alpha_k > 0$. Then, $\alpha^p x^p + \alpha_k x_k \leq \beta$ cuts off $\bar{x}$.

Proof: Since, as $\alpha^{p_i} x^{p_i} \leq \beta$ is tight at $\bar{x}$, we have $\alpha^{p_i} \bar{x}^{p_i} = \beta$. Since $\alpha_k > 0$ and $0 < \bar{x}_k < 1$, we have $\alpha_k \bar{x}_k > 0$. Thus, $\alpha^{p_i} \bar{x}^{p_i} + \alpha_k \bar{x}_k > \beta$ and $\bar{x}$ violates valid inequality $\alpha^p x^p + \alpha_k x_k \leq \beta$.

This theorem implies that we can generate cuts by lifting subproblem faces that are tight at the current fractional solution $\bar{x}$. These cuts can be incorporated in RMP without changing the subproblem structure. Note that Theorem 6.1 guarantees that a cut can be generated in this way only if it is possible to lift the face successfully (i.e., $\exists$ x_k such that $0 < \bar{x}_k < 1$ and $\alpha_k > 0$).

Lifting is a prevalent concept in the literature. Our contribution here is in providing a mechanism for identifying and generating potential inequalities to be lifted. For each subproblem, we apply FGP to identify faces (facets) of the corresponding polytope that are tight at the current DWD fractional solution $\bar{x}$. Having identified an inequality representing a subproblem face (facet), we attempt to lift it over variables associated with other subproblems in order to generate cuts successfully. Next, we describe FGP within this context.

6.3.1 Facet generation procedure

Gadidov et.al. [102] introduced FGP, which identifies a facet of a full-dimensional integer polytope $P \in R^n$ by separating a given fractional point $f^* \notin P$. This procedure relies on an oracle to solve an optimization problem over P. They embedded FGP within a B&B framework and generated cuts derived from facets of underlying knapsack polytopes. Here, we adapt FGP to identify whether a given fractional point f^* is an interior point or an inner point relative to a subproblem polytope. Further, if f^* is an inner point, FGP provides a face (facet) containing f^*. We describe this adaptation of FGP below.

Objective: Identify if a fractional point f^* is an inner point or an interior point w.r.t. to an underlying full-dimensional polytope $P \in R^n$. In case f^* is an inner point, identify a face of P containing f^*.

FGP Assumptions:

(A1) $P \in R^n$ is a full dimensional polytope and $0 \in P$.

(A2) $f^* \in P$.

(A3) There exists a set E_1 of n vectors representing linearly independent extreme points of P such that f^* belongs to the convex cone generated by E_1.

(A4) There exists an oracle to solve an integer program over P.

FGP solves the following LP problem to optimality using column generation:

$$z^* = \text{Min} \sum_{i \in Ext(P)} \alpha_i \tag{6.9}$$

s.t.

$$\sum_{i \in Ext(P)} \alpha_i \, x_i = f^* \tag{6.10}$$

$$\alpha_i \geq 0 \; i \in Ext(P), \tag{6.11}$$

where $Ext(P)$ represents the set of extreme point of the polytope P. The optimal solution z^* provides the following information:

- If $z^* = 1$, then $f^* \in P$ and f^* is an inner point.
- If $z^* < 1$, then $f^* \in P$ and f^* is an interior point.

For the case $z^* = 1$, the face (facet) containing f^* is generated by constructing the affine hull of the extreme points corresponding to the optimal basis. Note that $z^* > 1$ implies $f^* \in P$ and violates (A2).

In our adaptation, we use $\bar{x}^p$ for some $p \in P$ as fractional point f^*. Our oracle corresponds to the solver for subproblem $p \in P$. In addition, we know a priori that $\bar{x}^p \in Q_p$, thus guaranteeing $z^* \leq 1$. Moreover, since $\bar{x}^p = \sum_{j \in J_p} \bar{\lambda}_{jp} x^{jp}$, the extreme points used in the representation of $\bar{x}^p$ are known a priori. This provides an initial basis for the FGP column generation problem. If the solution to (6.9) - (6.11) gives $z^* < 1$, then $\bar{x}^p$ is

an interior point and we conclude that no subproblem face (facet) is tight at $\bar{x}^p$. However, if $z^* = 1$, $\bar{x}^p$ is an inner point and we construct the affine hull of the extreme points corresponding to the optimum basis of (6.9) - (6.11) to generate the valid inequality representing the subproblem face (facet) containing $\bar{x}^p$. Note that a degenerate optimal basis would imply that the inner point $\bar{x}^p$ lies on a face of dimension less than that of a facet and that FGP identifies the corresponding face. In the next section we describe our lifting step.

6.3.2 Lifting subproblem faces

After having identified subproblem faces that are tight at $\bar{x}^p$, we attempt to lift each sequentially over variables associated with other sub-problems. Based on Theorem 6.1, we need only to lift with respect to variables that are fractional in the current solution. Our lifting problem is an integer problem and can be solved using the same B&P scheme used for the original problem. We now describe the lifting problem.

Assume that FGP has identified face $\alpha^{p_j} x^{p_j} \leq \beta$ of the polytope associated with subproblem p_j, $j \in \{1,...,|P|\}$. Let T be the set of the decision variables to be lifted: $T = \{x_i \mid 0 < \bar{x}_i < 1, i \notin p_j\}$. We do not need to lift with respect to all variables in T. In fact, the lifting process can be terminated at any iteration after having obtained at least one positive coefficient for a lifted variable since such an inequality represents a cut. At iteration k, let $L_{k-1} \subset T$ represent the set of variables already lifted and let $x_k \in T \setminus L_{k-1}$ be the next variable to be lifted. The lifting problem is represented as follows:

$$z_k = Max \;\; \alpha^{p_j} x^{p_j} + \sum_{j \in L_{k-1}} \alpha_j x_j$$

s.t.

$$A_1 x^1 + A_2 x^2 + \cdots + A_{|P|} x^{|P|} \le b$$

$$D_p x^p \le d^p \qquad \forall p \in P \tag{6.12}$$

$$x^p \in R^{n_p} \qquad \forall p \in \{1,\ldots,|P|\}$$

$$x_k = 1.$$

The lifting coefficient for x_k is obtained from $\alpha_k = \beta - z_k$, where β is the right hand side of the inequality representing the face to be lifted. According to Theorem 6.1, if $\alpha_k > 0$ for any $x_k \in T$, a cut has been generated successfully. In the next section we illustrate an example of our lifting technique:

6.3.3 Example of lifting

We demonstrate our cutting plane methodology in application to the vertex-disjoint B&P approach for MWISP (see Chapter III). Figure 10 depicts a graph G and a vertex-disjoint partitioning of G into two subgraphs.

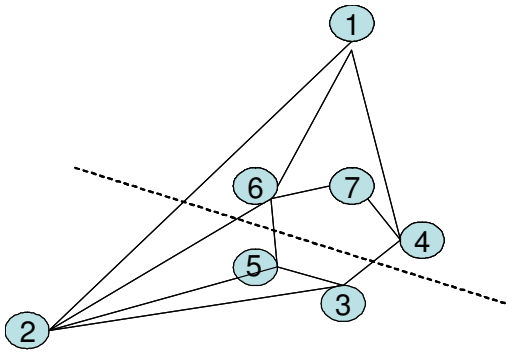
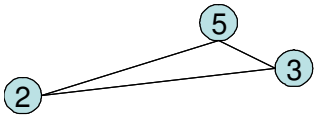
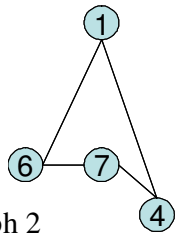


Figure 10: A graph and a vertex-disjoint partition into two subgraphs



Subgraph 1

Figure 11a: Vertex disjoint subgraph 1



Subgraph 2

Figure 11b: Vertex disjoint subgraph 2

The corresponding formulation for MWISP is as follows:

$$\left.\begin{aligned} x_1 + x_2 &\le 1 \\ x_2 + x_6 &\le 1 \\ x_6 + x_5 &\le 1 \\ x_3 + x_4 &\le 1 \end{aligned}\right\}$$ constraints that will be relegated to the master problem

$$\left.\begin{aligned} x_2 + x_5 &\le 1 \\ x_2 + x_3 &\le 1 \\ x_5 + x_3 &\le 1 \end{aligned}\right\}$$ constraints that will be used to form subproblem 1

$$\left.\begin{aligned} x_1 + x_6 &\le 1 \\ x_1 + x_4 &\le 1 \\ x_4 + x_7 &\le 1 \\ x_7 + x_6 &\le 1 \end{aligned}\right\}$$ constraints that will be used to form subproblem 2

$$x_1,\ x_2,\ x_3,\ x_4,\ x_5,\ x_6,\ x_7 \in \{0,1\}.$$

The optimal RMP solution at the root node is fractional:

$\underline{x}_1 = \underline{x}_2 = \underline{x}_4 = \underline{x}_5 = \underline{x}_6 = \underline{x}_7 = 0.5, \underline{x}_3 = 0$, and $Z^*_{RMP_{VD}} = 3.0$.

A minimal description of Q_1 for subproblem $p = 1$ (corresponding to subgraph 1 in Figure 11a) is as follows:

$$\begin{aligned} x_2 + x_5 &\leq 1 \\ x_2 + x_3 &\leq 1 \\ x_5 + x_3 &\leq 1 \\ x_2 + x_3 + x_5 &\leq 1 \\ 0 \leq x_2, x_3, x_5 &\leq 1 \end{aligned} \tag{6.13}$$

Similarly, a minimal description of Q_2 for subproblem $p = 2$ (corresponding to subgraph 2 in Figure 11b) is as follows:

$$\begin{aligned} x_1 + x_6 &\leq 1 \\ x_1 + x_4 &\leq 1 \\ x_4 + x_7 &\leq 1 \\ x_7 + x_6 &\leq 1 \\ 0 \leq x_1, x_4, x_6, x_7 &\leq 1 \ . \end{aligned} \tag{6.14}$$

Note that subproblem 1 face $x_2 + x_5 \leq 1$ is tight at the current fractional solution. Lifting this face over the decision variables associated with subproblem 2 entails lifting with respect to variables related to vertices 1, 6, and 4. Note that the variable related to vertex 7 need not be lifted since it is not adjacent to either vertex 2 or 5. Moreover, variables x_1, x_4, x_6 are all fractional; we can lift with respect to each one of them. Lifting produces zero coefficients $\alpha_1 = 0$ and $\alpha_4 = 0$ for x_1 and x_4; however, it

produces a positive coefficient $\alpha_6 = 1$ for variable x_6, generating the clique-inequality $x_2 + x_5 + x_6 \leq 1$. Note that this inequality cuts off the current fractional solution ($\underline{x}_2 + \underline{x}_5 + \underline{x}_6 = 1.5 > 1$) ! Thus, we have successfully generated a cut using our scheme. The next section describes our computational tests.

6.4 Computational evaluation

We evaluate our cut-generation methodology in application to the vertex-disjoint B&P approach for MWISP (see Chapter III), focusing on instances from the Second DIMACS Implementation Challenge (we actually use the complements of the listed graphs). Table 12 describes each test instance and presents results. The first five columns specify the instance, the associated number of vertices $|V|$, the number of edges $|E|$, the % Density Δ, and the number of partitions P used in the vertex-disjoint scheme.

For each instance we generate one round of cuts at the root node. If an RMP solution is fractional, we use FGP in an attempt to identify a face (facet) from each subproblem that is tight at the fractional solution. Each such face is then lifted in an attempt to generate a cut. If a cut is identified, we incorporate it in RMP, which is then reoptimized. The bound obtained after one iteration of cut generation is denoted $Z_{LP}(1)$. To evaluate the tightness obtained from cut generation, we compare $Z_{LP}(1)$ (column 8) with the optimal root node solution of RMP, $Z_{LP}(0)$ (column 6). Columns 7 and 9 record run times to obtain $Z_{LP}(0)$ and $Z_{LP}(1)$, respectively.

Results show that our cut generation methodology improves the bound obtained in 9 of these 11 instances; it did not identify a valid cut for 2 instances. The failure in

both of these cases was because lifting the subproblem faces did not yield a positive coefficient for any fractional variable. Our results indicate that substantial computational effort may be required to generate a cut using this approach. Lifting entails most of the computational effort because it involves solving an integer program for each lifted variable. We lifted each identified subproblem face over all decision variables in the associated set T. Thus, the computational effort we report is related to both the number of partitions $|P|$ and the number of vertices $|\hat{V}|$. Finally, the improvement in the bound at the end of just one round of cut generation is not substantial. It is possible that additional rounds of cuts could yield additional, tighter bounds. However, there is no guarantee that our approach will generate the deepest possible cuts.

6.5 Conclusions

In this chapter we present a new technique for generating cuts within B&P. We also provide a preliminary computational evaluation. Although lifting is a prevalent technique, our contribution is in providing a mechanism for identifying potential inequalities to be lifted within B&P to yield cuts that do not destroy subproblem structure.

We have three primary concerns. First, we are unable to guarantee that the identified subproblem face will yield a cut when lifted. Second, the cut obtained is not guaranteed to be the deepest cut possible (which affects the % improvement achieved in the bound). Third, lifting may require a prohibitive run. While the first and the second concerns are inherent and cannot be resolved, there is scope for reducing the computational effort. Specifically, we can avoid lifting all the relevant decision variables

and terminate upon achieving a positive coefficient for any lifted variable. We could also explore problem-specific techniques for identifying a priori decision variables which will yield a cut upon lifting. Further, we need to evaluate our approach in other IP applications besides MWISP. Our future research will be directed along these lines.

Table 12
Preliminary results for lifting

Instance	|V|	|E|	Δ	P	$Z_{LP}(0)$	Time	$Z_{LP}(1)$	Time
MANN_a9	45	72	7.3	5	18	1.781	18	5.945
johnson8-4-4	70	560	23.2	6	16.5	0.452	16.1684	35.431
johnson16-2-4	120	1680	23.5	10	10.5	2.046	9.25	91.421
keller4	171	5100	35.1	4	17.7533	14.171	17.5842	520.089
hamming8-4	256	11776	36.1	5	20.827	43.999	20.7556	3984.37
brock200-3	200	7852	39.5	4	27.5205	40.484	27.2219	1280.76
johnson8-2-4	28	168	44.4	5	5.25	0.233	4.375	6.978
c-fat-2005	200	11.427	57.4	5	66.667	8.906	66.667	6591.31
p_hat300-1	300	33.917	75.6	2	12.867	533.14	12.85	2836.19
c-fat-2002	200	16665	83.7	4	26.5	20.593	26.3846	761.766
c-fat-2001	200	18366	92.3	3	14	41.342	13.8	859.062

CHAPTER VII

CUT GENERATION WITHIN B&P – INVOKING LIFT & PROJECT

7.1 Introduction

DWD reformulation can provide a tighter bound than that given by the LP relaxation of a model. However, in a typical implementation of B&P, the bounds obtained may not tight enough to solve challenging instances effectively. B&P can potentially be enhanced by incorporating cutting planes to form a branch-and-cut-and-price (BCP) approach. However, incorporating cutting planes within B&P is challenging. Chapter VI mentions the challenges involved in implementing traditional cutting plane methods within a B&P framework. Further, it presents a lifting technique for generating cutting planes in terms of the original problem variables. However, this technique suffers several drawbacks. Primarily, there is no guarantee that the lifting technique can identify a cut. Moreover, any cut obtained from lifting is not guaranteed to be the deepest. Addressing these concerns, we now explore generic (i.e., without relying on the polyhedral properties of the underlying problem) cutting plane methods within a B&P framework. We emphasize that, although we implement it specifically for MWISP, our approach is useful in generic applications of B&P.

As in Chapter VI, our approach relies on identifying faces (facets) of a subproblem polytope using a modification of FGP [102]. We emphasize that resulting valid inequalities are of no use if incorporated in RMP, since DWD reformulation invokes them implicitly as it optimizes over the integer convex hull of each subproblem. We begin by presenting a theoretical framework for generating valid cutting planes in a Chvatal-Gomory (C-G) fashion by combining faces (facets) generated from the

subproblems in conjunction with master problem inequalities. However a practical implementation of the C-G cut relies on identifying the C-G multipliers which is not straightforward. We overcome this challenge by exploring the L&P technique and show how to invoke L&P cuts within a B&P framework. This is the basis of our cut generation scheme, which we evaluate through computational tests. Our goal is to present a framework for generating generic cutting planes within the B&P approach.

This chapter has five sections. The B&P formulations discussed in Chapter VI are referenced instead of duplicating them here. In section 7.2 we present the insight for generating a C-G cut within B&P. Section 7.3 presents L&P with respect to DWD reformulation while section 7.4 describes our cut generation scheme. Section 7.5 presents our computational tests.

7.2 Cut generation scheme

Let $\bar{x}$ ($[\bar{x}^1, \bar{x}^2, \ldots, \bar{x}^{|P|}]^T$) denote the current DWD fractional solution obtained by the transformation $\bar{x}^p = \sum_{j \in J_p} \bar{\lambda}_{jp} x^{jp}$ where $\bar{\lambda}$ is the vector representing the optimal basic variables in RMP. From the principles of DWD, we know that $\bar{x}$ corresponds to an optimal feasible basis for DWD" (see Chapter VI). This implies that, at $\bar{x}$, a subset of constraints (6.7) and (6.8) is tight. Note that, while it is easy to determine the subset of constraints (6.7) that are tight at $\bar{x}$, we can not do the same for constraints (6.8) because we don't know them explicitly. Moreover, there could be exponential number of constraints in set (6.8). However, if constraint set (6.8) were available explicitly, we could generate a C-G cut by taking a combination of selected constraints in (6.7) and (6.8). In the next section we illustrate an example of such a procedure.

7.2.1 Example of a C-G cut in B&P

We demonstrate an example based on the vertex-disjoint B&P approach for MWISP. We refer to Figure 10 from Chapter VI, which depicts a graph and a vertex-disjoint partition into two subgraphs. We also refer to the corresponding formulations from Chapter VI.

The optimal RMP solution at the root node using the vertex-disjoint B&P scheme is fractional:

$$\underline{x}_1 = \underline{x}_2 = \underline{x}_4 = \underline{x}_5 = \underline{x}_6 = \underline{x}_7 = 0.5 \text{ and } \underline{x}_3 = 0 \text{ and } Z^*_{RMP} = 3.0.$$

Referring to the DWD" reformulation for this example (Chapter VI), we derive a C-G cut using a linear combination of the following inequalities:

$x_2 + x_6 \leq 1$ (Master constraint)

$x_6 + x_5 \leq 1$ (Master constraint)

$x_2 + x_3 + x_5 \leq 1$ (face of subproblem polytope $p = 1$).

Using a coefficient of 0.5 for each constraint, a linear combination results in

$$x_2 + 0.5\, x_3 + x_5 + x_6 \leq 1.5$$

Integer rounding generates the clique inequality $x_2 + x_5 + x_6 \leq 1$. Note that this inequality cuts off the current fractional solution ($\underline{x}_2 + \underline{x}_5 + \underline{x}_6 = 1.5 > 1$)! The next section summarizes our C-G scheme for B&P.

7.2.2 Implementing C-G cuts within B&P

Our approach is based on the fact that $\bar{x}$ corresponds to an optimal feasible basis for DWD". We seek to identify the set of hyperplanes that are tight at $\bar{x}$. This set comprises a subset of master problem constraints and a subset of subproblem faces

(facets). We use FGP (see section 6.3) to identify the subproblem faces that are tight at $\bar{x}$. For each subproblem, we apply FGP to identify faces (facets) of the corresponding polytope that are tight at the current DWD fractional solution $\bar{x}$. A C-G cut can then be derived by taking a linear combination of the identified subproblem faces (facets) and the master problem constraints that are tight at $\bar{x}$.

However, the challenge involved is in identifying the multipliers for the linear combination used to derive the C-G cut. Although our simple example demonstrates the feasibility of invoking C-G cuts within B&P theoretically, a practical implementation is not straightforward. In the next section we present a theoretical framework for using L&P within B&P while addressing our concerns about C-G cuts.

7.3 Lift & project

L&P tightens the linear relaxation of an integer program by lifting it into a higher dimensional space where a tighter formulation is obtained. This higher dimension polyhedron, when projected back onto the original space, provides a tighter approximation of the integer convex hull [6, 7, 114]. L&P utilizes this higher dimension polyhedron to derive strong cutting planes for the original polyhedron [6, 7]. In the next section we present our L&P scheme for B&P.

7.3.1 Lift & project using the binary disjunction

We begin by invoking L&P for DWD". Recall that the DWD" reformulation is obtained by explicitly invoking the integer convex hull of each subproblem polytope. Assuming that a DWD" reformulation is available, we show how to invoke L&P. Later,

we address the concern that the DWD" reformulation is not explicitly available. We start with the following 0-1 program, which corresponds to the DWD" reformulation:

$$\max \quad \sum_{p\in P} c^p x^p \tag{7.1}$$

s.t.

$$\sum_{p\in P} A_p x^p \le b \tag{7.2}$$

$$\tilde{D}_p x^p \le \tilde{d}^p \ \forall p \in P \tag{7.3}$$

$$x^p \in B^{n_p}, \forall p \in \{1,\ldots,|P|\}, \tag{7.4}$$

where A_p represents the matrix of coefficients associated with master problem constraints, $\tilde{D}_p$ represents the matrix of coefficients associated with constraints representing the integer convex hull of subproblem polytope $p \in P$, $x^p \in B^{n_p}$ is the vector of binary decision variables associated with subproblem $p \in P$, and $c^p \in R^{n_p}$ is the corresponding vector of cost coefficients.

The corresponding linear relaxation is obtained by relaxing binary restriction (7.4). A disjunctive relaxation of (7.2) - (7.4) is obtained by imposing the 0-1 disjunction on a single variable $x_j \in \{0,1\}$ as in $(x_j \le 0) \vee (x_j \ge 1)$. The conjunctive normal form for the disjunctive set is represented as

$$\begin{aligned} &\sum_{p\in P} A_p x^p \le b \\ &\tilde{D}_p x^p \le \tilde{d}^p \quad \forall p \in P \\ &(x_j \le 0) \vee (x_j \ge 1) \\ &x^p \in R^{n_p} \quad \forall p \in \{1,\ldots,|P|\}, \end{aligned} \tag{7.5}$$

and the disjunctive normal form is represented as

$$\begin{array}{l} \sum_{p\in P} A_p x^p \le b \\ \tilde{D}_p x^p \le \tilde{d}^p \ \forall p \in P \\ \left(x_j \le 0\right) \\ x^p \in R^{n_p}, \forall p \in \{1,\ldots,|P|\} \end{array} \quad \vee \quad \begin{array}{l} \sum_{p\in P} A_p x^p \le b \\ \tilde{D}_p x^p \le \tilde{d}^p \ \forall p \in P \\ \left(x_j \ge 1\right) \\ x^p \in R^{n_p}, \forall p \in \{1,\ldots,|P|\} \end{array} \qquad (7.6)$$

7.3.2 Compact representation of the convex hull

L&P invokes the convex hull representation of the union of two polyhedra, each of which corresponds to a disjunctive set [6, 7]. Let $\tilde{H}$ denote this closed convex hull. $\tilde{H}$ is thus the set of points $x^p \in R^{n_p}\ \forall p \in P$ for which there exist vectors $\left(y_0 \in R^1, y^p \in R^{n_p} \forall p \in P\right)$ and $\left(z_0 \in R^1, z^p \in R^{n_p} \forall p \in P\right)$ such that

$$\begin{array}{ll} x^p = y^p + z^p & \forall p \in P \\ y_0 + z_0 = 1 & \\ \sum_{p\in P} A_p y^p \le b y_0 & \\ \tilde{D}_p y^p \le \tilde{d} y_0 & \forall p \in P \\ y_j = 0 & \\ \sum_{p\in P} A_p z^p \le b z_0 & \\ \tilde{D}_p z^p \le \tilde{d} z_0 & \forall p \in P \\ z_j - z_0 = 0 & \\ y^p, z^p \in R^{n_p}. & \end{array} \qquad (7.7)$$

L&P exploits the observation that $\bar{x} \notin \tilde{H}$ and, thus, facets of $\tilde{H}$ can be used to cut off $\bar{x}$ [6, 7]. L&P uses the reverse polar $\tilde{H}^*$ of $\tilde{H}$ in order to generate these cuts:

$$\tilde{H}^* = \begin{cases} \alpha^1 \in R^{n_1}, \alpha^2 \in R^{n_2}, \beta \in R^1 : \\ \alpha \le uA + \mu\tilde{D} + u_0 e_j \\ \alpha \le vA + \upsilon\tilde{D} - v_0 e_j \\ \beta \ge ub + \mu\tilde{d} \\ \beta \ge vb + \upsilon\tilde{d} - v_0 \\ \mathrm{u}, \mu, \mathrm{v}, \upsilon, \mathrm{u}_0, v_0 \ge 0, \end{cases}$$

where $A = \begin{bmatrix} A_1 & A_2 & \cdots & A_P \end{bmatrix}$, $\tilde{D} = \begin{bmatrix} \tilde{D}_1 & 0 & \cdots & 0 \\ 0 & \tilde{D}_2 & \cdots & 0 \\ \cdots & \cdots & \ddots & 0 \\ 0 & 0 & & \tilde{D}_P \end{bmatrix}$, and $\tilde{d} = \begin{bmatrix} \tilde{d}_1 \\ \tilde{d}_2 \\ \vdots \\ \tilde{d}_P \end{bmatrix}$

If $\tilde{H}$ is full-dimensional, inequality $\sum_{p \in P} \alpha^p x^p \le \beta$ defines a facet of $\tilde{H}$ IFF $(\alpha^1, \alpha^2, \cdots, \alpha^p, \beta)$ is an extreme ray of $\tilde{H}^*$ [6, 7]. In order to generate the extreme rays of $\tilde{H}^*$, L&P solves a linear program over a normalized version of the cone $\tilde{H}^*$ [6, 7]. This linear program identifies the facet of $\tilde{H}$ that is most violated by the current fractional point $\bar{x} = (\bar{x}^1, \bar{x}^2, \ldots \bar{x}^P)$ and is called the cut generating linear program (CGLP) [6, 7]:

$$\text{Min} \left(\beta - \sum_{p \in P} \alpha^p \bar{x}^p \right)$$

s.t.

$$\begin{aligned} &\alpha \le uA + \mu\tilde{D} + u_0 e_j \\ &\alpha \le vA + \upsilon\tilde{D} - v_0 e_j \\ &\beta \ge ub + \mu\tilde{d} \\ &\beta \ge vb + \upsilon\tilde{d} - v_0 \\ &\text{-}1 \le \beta \le 1 \\ &\mathrm{u}, \mu, \mathrm{v}, \upsilon, \mathrm{u}_0, v_0 \ge 0 \end{aligned} \qquad \text{(CGLP)}$$

In CGLP, we use a normalization that restricts $\beta \in [0,1]$, allowing us to deal with a polyhedron instead of the cone $\tilde{H}^*$. Different normalization forms can be used [6, 7] but care must be taken to assure that the normalization does not distort the cone; that is, extreme points of the CGLP polyhedron should still correspond to facets of $\tilde{H}$. Note that we need not solve CGLP to optimality; we can stop anytime the objective function value is positive. However, this would not guarantee the deepest cut.

7.4 Generating L&P cuts within B&P

In section 7.3 we showed that a L&P cut can be derived assuming that the DWD" reformulation is explicitly available. However, DWD" is not explicitly available. Moreover, even if it were available, the minimal representation could involve a large number of facets, so that solving CGLP with all facets invoked explicitly could take a prohibitive amount of time.

We overcome these challenges by posing CGLP as a column generation problem in which each generated column is associated with a facet of a subproblem polytope. The next section describes our column generation framework in more detail.

7.4.1 Solving CGLP using column generation: master problem

Our approach eliminates the need for a priori explicit information defining all facets of each subproblem polytope. Instead, we use a column generation scheme to identify required facets dynamically. We emphasize that CGLP is a linear program and we use a column generation approach to solve it. Moreover, our column generation approach does not invoke DWD but is a Type II column generation [130], analogous to that used for cutting stock problems [59]. Our column generation scheme entails solving

a RMP comprising columns representing master problem constraint set A and a subset of the subproblem facets $\tilde{D}$. The dual solution provided by RMP is used by an oracle to generate improving columns corresponding to subproblem facet defining equalities, which are entered into RMP in the subsequent Simplex iteration. Optimality for CGLP is achieved when no improving column is identified by the oracle. We now present RMP for CGLP:

$$Z_{CGLP} = \text{Min}\left(\beta - \sum_{p\in P}\alpha^p \bar{x}^p\right) \tag{7.8}$$

s.t.

$$\alpha \le uA + \mu'\tilde{D}' + u_0 e_j \tag{7.9}$$

$$\alpha \le vA + \upsilon'\tilde{D}' - v_0 e_j \tag{7.10}$$

$$\beta \ge ub + \mu'\tilde{d}' \tag{7.11}$$

$$\beta \ge vb + \upsilon'\tilde{d}' - v_0 \tag{7.12}$$

$$-1 \le \beta \le 1 \tag{7.13}$$

$$\text{u}, \mu', \text{v}, \upsilon', \text{u}_0, v_0 \ge 0 \tag{7.14}$$

where, μ' and υ' correspond to the subset of oracle-prescribed (i.e., generated) columns that populate the current RMP. In CGLP, columns associated with decision variables μ and υ correspond to facets of subproblem polytopes, while those associated with decision variables u and v correspond to the master problem constraints. Since master problem constraints are known, we initiate RMP for CGLP with columns corresponding to decision variables u and v. Optimal dual values from the solution of RMP are used by the oracle to generate columns corresponding to decision variables μ and υ.

Our column generation scheme for CGLP need not be solved to optimality and can be stopped anytime the objective function value is greater than zero. We next describe the pricing problem used for generating columns corresponding to facets of subproblem polytopes.

7.4.2 Solving CGLP using column generation: pricing subproblem

Since each generated column is associated with a facet of a subproblem polytope, each pricing problem is an IP over the polar of a subproblem polytope. We use δ, η, δ_0, and η_0, the optimal solution to the dual of RMP ((7.8) – (7.14)), to price out columns μ and υ. μ', υ', δ and η are partitioned to correspond with the subproblems in the forms

$$\mu' = \begin{bmatrix} \mu_1' \\ \mu_2' \\ \vdots \\ \mu_P' \end{bmatrix}, \ \upsilon' = \begin{bmatrix} \upsilon_1' \\ \upsilon_2' \\ \vdots \\ \upsilon_P' \end{bmatrix}, \ \delta = \begin{bmatrix} \delta_1 \\ \delta_2 \\ \\ \delta_P \end{bmatrix} \text{ and } \eta = \begin{bmatrix} \eta_1 \\ \eta_2 \\ \\ \eta_P \end{bmatrix},$$

respectively. We denote the polar of $Q_p = \left\{ x^p \in B^{|n_p|} : D_p x^p \le d^p \right\}$ by Q_p^*. Pricing subproblems $SP(\delta_p, \delta_0)$ and $SP(\eta_p, \eta_0)$ have the following forms:

$$SP(\delta_p, \delta_0) : Z^*(\delta_p, \delta_0) = \text{Max } \{ -\delta_p \ y_p + \delta_0, \text{ where } y_p \in Q_p^* \}. \tag{7.15}$$

and

$$SP(\eta_p, \eta_0) : Z^*(\eta_p, \eta_0) = \text{Max } \{ -\eta_p \ y_p + \eta_0, \text{ where } y_p \in Q_p^* \}. \tag{7.16}$$

Both these pricing subproblems are integer programs over the polar Q_p^*. A column obtained by solving $SP(\delta_p, \delta_0)$ (or $SP(\eta_p, \eta_0)$) corresponds to extreme point of Q_p^* and, thus, represents a facet of Q_p. A column obtained by solving $SP(\delta_p, \delta_0)$ (or $SP(\eta_p, \eta_0)$) is considered improving if $Z^*(\delta_p, \delta_0) > 0$ (or $Z^*(\eta_p, \eta_0) > 0$). At the optimal solution,

no improving column can be identified by any subproblem. In the next section we describe an approach to solve the pricing problem.

7.4.3 Solving the pricing problem

We begin by giving a theorem that provides insight into FGP (see Chapter VI):

Theorem 7.1 Solving an IP over the polar Q_p^* is dual to solving FGP on Q_p.

Proof: We begin by stating the FGP problem on Q_p:

Primal FGP(Q_p): Min $\sum_{i \in Ext(P)} \alpha_i$

s.t.

$$\sum_{i \in Ext(P)} \alpha_i \, x_i = f^* \qquad \text{(Primal)}$$

$$\alpha_i \geq 0 \;\; i \in Ext(Q_p)$$

The dual to FGP(Q_p) is given by

Dual {FGP(Q_p)}: Max $f^* \, y^T$

s.t.

$$y^T \, x_i \leq 1 \; \forall i \in Ext(Q_p)$$

y^T free

Assuming that Q_p is a bounded polytope containing the origin, we know from Theorem 9.1, Schrijver 1986 [111] that the set $\{ y^T \in R^{n_p} \mid y^T \, x_i \leq 1 \; \forall i \in Ext(Q_p) \}$ represents the polar Q_p^* of Q_p. Thus, the linear programming dual to FGP(Q_p) can be represented as

Dual {FGP(Q_p)}: Max $f^* \, y^T \mid y^T \in Q_p^*$,

which corresponds to solving an IP over the polar Q_p^*. [QED].

Theorem 7.1 shows that, in the column generation framework, solving the pricing problem is equivalent to solving FGP(Q_p) with f^* corresponding to $-\delta_p$ (or $-\eta_p$). The next section describes preliminary computational tests of our approach.

7.5 Computational benchmarks

We apply our cut generation scheme to the vertex-disjoint formulation of MWISP discussed in Chapter III. We focus our preliminary tests on instances from the Second DIMACS Implementation Challenge (we actually use the complements of the listed graphs). Table 7.1 describes each test instance and presents results. The first five columns specify the instance, the associated number of vertices, $|V|$; the number of edges, $|E|$; the % Density, Δ; and the number of partitions, P; used in the vertex-disjoint scheme.

For each instance we generate ten rounds of cuts at the root node. If an RMP solution is fractional, we select the most fractional variable and generate an L&P cut using a disjunction based on this fractional variable. We solve CGLP to optimality for each such iteration to optimality. The resulting cut is then incorporated in RMP, which is then reoptimized. The bound obtained after 10 such iterations is denoted $Z_{LP}(10)$. To evaluate the tightness obtained from cut generation, we compare $Z_{LP}(10)$ (column 7) with the optimal root node solution of RMP, $Z_{LP}(0)$ (column 6).

Table 13 shows (columns 6 and 7) that cut generation improves the bound obtained in each of the 11 instances. Unlike the lifting scheme presented in Chapter V, solving CGLP to optimality guarantees that L&P prescribes the deepest cut. In addition, our L&P scheme is guaranteed to generate a cut.

Our main concern is the time consumed to obtain each cut – especially for larger instances. One option for reducing run time is to terminate CGLP as soon as a cut is obtained, rather than solving it to optimality. More time can then be spent on additional cut-generating iterations, compensating for the depth of the cut obtained on each iteration. A second improvement for larger instances would involve invoking only the RMP constraints that are tight instead of all RMP constraints, thus reducing the size of CGLP. Finally, recent research has shown that a L&P cut can be generated from the optimal Simplex tableau without solving CGLP, thus reducing run time substantially. A similar technique within the B&P framework could be explored. Our research continues along these lines.

Table 13

Results of Lift & Project Cut generation scheme within B&P

Instance	**|V|**	**|E|**	**Δ**	**P**	$Z_{LP}(0)$	$Z_{LP}(10)$
MANN_a9	45	72	7.3	5	18	17.33
johnson8-4-4	70	560	23.2	6	16.5	15.9853
johnson16-2-4	120	1680	23.5	10	10.5	10.0
keller4	171	5100	35.1	4	17.7533	17.2579
hamming8-4	256	11776	36.1	5	20.827	20.2495
johnson8-2-4	28	168	44.4	5	5.25	4.9167
p_hat300-1	300	33.917	75.6	2	12.867	12.8286
c-fat-2002	200	16665	83.7	4	26.5	25.3623
c-fat-2001	200	18366	92.3	3	14	13.9022

CHAPTER VIII

CONCLUSION AND FUTURE RESEARCH

In this research, we have explored B&P approaches for solving MWISP, one of the most well-known and well-studied NP-hard problems in the field of combinatorial optimization. In the first part of this research, we explored vertex and edge-disjoint decompositions of the underlying graph to develop B&P approaches for MWISP. We demonstrated that vertex-disjoint partitioning scheme gives an effective approach for relatively sparse graphs (i.e., density less than 30%). We showed that the edge-disjoint approach is less effective than the vertex-disjoint scheme because the associated DWD reformulation of the latter entails a slow rate of convergence. Further research can explore avenues for enhancing the effectiveness of the edge-disjoint approach for MWISP. Also, future research can explore methods for determining an optimal partitioning of a graph for MWISP. An ideal partitioning should yield an optimal integer solution at the root node of the B&B tree. However, this does not appear practical. A more realistic goal would be to identify optimal partitionings for both vertex and edge-disjoint approaches with the goal of minimizing the run time required to prescribe an optimal integral solution.

In the second part of this research, we addressed convergence properties of DWD. We described available techniques for improving the rate of convergence and presented preliminary research towards exploring non-parametric approaches for stabilizing DWD. We also demonstrated our efforts for improving the rate of convergence associated with the edge-disjoint B&P approach. Future research can explore more generic stabilization

techniques, especially for challenging set partitioning problems. Also, future research can continue to explore non-parametric approaches for stabilizing DWD.

In the third part of this research, we explored more fundamental concepts towards enhancing the strength of B&P as a useful integer programming tool. A primary challenge posed in B&P is in generating cuts that do not distort subproblem structure. Traditional implementations of C-G and L&P cuts can not be successful within a B&P framework. We presented two new methodologies for generating *generic* cutting planes within the B&P framework. The first methodology generates cuts by using FGP to identify faces (facets) of subproblem polytopes and lifting associated inequalities; the second methodology computes L&P cuts within B&P. We successfully demonstrated the feasibility of our approaches and presented preliminary computational tests of each. Future research can focus on devising more effective methods to implement the proposed cut-generation approaches with the ultimate goal of building a generic Branch-and-Price-and-Cut framework.

REFERENCES

[1] I. Adler, A. Ülkücü, On the number of iterations in Dantzig-Wolfe Decomposition Algorithm, in: Decomposition of Large Scale Problems, D.M. Himmelblau (Ed.), North-Holland, Amsterdam, 1972, pp. 181-187.

[2] E. Balas, Intersection cuts: A new type of cutting planes for integer programming, Operations Research. 19 (1971) 19-39.

[3] E. Balas, Disjunctive programming, Annals of Discrete Mathematics. 5 (1979) 3-51.

[4] E. Balas, Disjunctive programming: Properties of the convex hull of feasible points, Discrete Applied Mathematics. 89 (1998) 1-44.

[5] E. Balas, S. Ceria, G. Cornuejols, N. Natraj, Gomory cuts revisited, Operations Research Letters. 19 (1996) 1-9.

[6] E. Balas, S. Ceria, G. Cornuejols, A Lift-and-Project cutting plane algorithm for mixed 0-1 programs, Mathematical Programming. 58 (1993) 295-324.

[7] E. Balas, S. Ceria, G. Cornuejols, Mixed 0-1 programming by Lift-and-Project in a Branch-and-Cut framework, Management Science. 42 (1996) 1229-1246.

[8] E. Balas, R. Jeroslow, Strengthening cuts for mixed integer programs, European Journal of Operations Research. 4 (1980) 224-234.

[9] E. Balas, M. Perregaard, Lift-and-project for mixed 0-1 programming: Recent progress, Discrete Applied Mathematics. 123 (2002) 129-154.

[10] E. Balas, M. Perregaard, A precise correspondence between Lift-and-Project cuts, simple disjunctive cuts, and mixed integer Gomory cuts for 0-1 programming, Mathematical Programming, Ser. B. 94 (2003) 221-245.

[11] E. Balas, J. Xue, Minimum weighted coloring of triangulated graphs with applications to maximum weight vertex packing and clique finding in arbitrary graphs, SIAM J Comp. 20 (1991), 209–221 [Addendum, SIAM J Comp. 21 (1992) 1000].

[12] E. Balas, J. Xue, Weighted and unweighted maximum clique algorithms with upper bounds from fractional coloring, Algorithmica. 15 (1996) 397–412.

[13] E. Balas, C.S. Yu, Finding a maximum clique in an arbitrary graph, SIAM J Comp. 15 (1986) 1054–1068.

[14] F. Barahona, R. Anbil, The volume algorithm: Producing primal solutions with a subgradient method, Mathematical Programming. 87 (2000) 385-399.

[15] C. Barnhart, C.A. Hane, P.H. Vance, Using Branch-and-Price-and-Cut to solve origin-destination integer multi-commodity flow problems, Operations Research. 48 (2) (2000) 318-326.

[16] C. Barnhart, E.L. Johnson, G.L. Nemhauser, M.W.P. Savelsbergh, P.H. Vance, Branch and Price: Column generation for solving huge integer programs, Operations Research. 46 (3) (1998) 316-329.

[17] M.S. Bazaraa, J.J. Jarvis, H.D. Sherali, Linear programming and network flows, Third edition, John Wiley & Sons, NewYork, 2005.

[18] E.M.L. Beale, Branch-and-bound methods for mathematical programming systems, in: Discrete Optimization, P.L. Hammer, E.L. Johnson, B.H. Korte (Eds.), Annals of Discrete Mathematics, Number 5, North-Holland, Amsterdam, 1979, pp. 201–219.

[19] E.M.L. Beale, Branch-and-bound methods for numerical optimization, in: M.M. Barritt, D.Wishart (Eds.), COMPSTAT 80: Proceedings in computational statistics, Physica-Verlag, Heidelberg, 1980, pp. 11–20.

[20] E.M.L. Beale, Integer programming, in compututational mathematical programming, in: K. Schittkowski (Ed.), Applications of Mathematical Programming, Springer-Verlag, Berlin, 1985, pp. 1–24.

[21] E.M.L. Beale, J.A. Tomlin, Special facilities in a general mathematical programming system for non-convex problems using ordered sets of variables, in: Proceedings of the Fifth International Conference on Operations Research, in: J. Lawrence (Ed.), 1970, pp. 447–454.

[22] N. Beaumont, An algorithm for disjunctive programming, European Journal Of Operations Research. 48 (1990) 362–371.

[23] H. Ben-Amor, J. Desrosiers, A. Frangioni, Stabilization in column generation, Les Cahiers du GERAD G-2004-62, Canada, 2004.

[24] H. Ben-Amor, J. Desrosiers, A Proximal Trust Region Algorithm for Column Generation Stabilization, Computers and Operations Research. 33 (2006) 910-927.

[25] P. Berman, A. Pelc, Distributed fault diagnosis for multiprocessor systems, in: Proceedings of the 20th Annual International Symposium on Fault-tolerant Computing, Newcastle, UK, 1990, pp. 340–346.

[26] I.M. Bomze, M. Budinich, P.M. Pardalos, M. Pelillo, The maximum clique problem, in: Handbook of Combinatorial Optimization, Kluwer Academic Publishers, Boston, MA, 1999, pp. 1–74.

[27] J. Bramel, D. Simchi-Levi, On the effectiveness of set covering formulations for the vehicle routing problem with time windows, Operations Research. 45 (1997) 295–301.

[28] O. Briant, P. Meurdesoif, K. Monneris, N. Perrot, F. Vanderbeck, C. Lemarechal, C. Tadonki, J.P. Vial, C. Beltran, Comparison of various approaches for column generation, Presented at the Eighth Aussois Workshop on Combinatorial Optimization, Aussois, France (2004a)

[29] O. Briant, P. Meurdesoif, N. Perrot, C. Lemarechal, S. Michel, and F. Vanderbeck, Comparison of Bundle Classical Column Generation, Presentation at the INFORMS/CORS Joint International Meeting, Banff, Alberta, Canada (2004b).

[30] D. Brélaz, New methods to color the vertices of a graph, Commun ACM. 22 (1979) 1–256.

[31] C. Bron, J. Kerbosch, Algorithm 457: Finding all cliques of an undirected graph, Commun ACM. 16 (1973) 575–577.

[32] S. Burer, R.D.C. Monteiro, Y. Zhang, Maximum stable set formulations and heuristics based on continuous optimization, Math Program Series A. 94 (2002) 137–166.

[33] R. Carraghan, P.M. Pardalos, An exact algorithm for the maximum clique problem, Operations Research Letters. 9 (1990) 375–382.

[34] J.M.V. de Carvalho, Using extra dual cuts to accelerate column generation, INFORMS Journal on Computing, 17 (2) (2005) 175-182.

[35] D. Cattrysse, M. Salomon, L.N. Van Wassenhove, A set partitioning heuristic for the generalized assignment problem, European Journal of Operational Research. 72 (1994) 167-174.

[36] V. Chvatal, Edmonds Polytopes and a hierarchy of combinatorial problems, Discrete Mathematics. 5 (1973) 305-337.

[37] V. Chvatal, On certain polytopes associated with graphs, Centre de Recherches Mathematiques -238, Universite de Montreal, Canada, 1972.

[38] K. Corradi, S. Szabo, A combinatorial approach for Keller's Conjecture, Period Math Hung. 21 (1990) 95–100.

[39] G.B. Dantzig, P. Wolfe, Decomposition principle for linear programs, Operations Research. 8 (1960) 101-111.

[40] G. Desaulniers, J. Desrosiers, M.M. Solomon, Accelerating strategies in column generation methods for vehicle routing and crew scheduling problems, in: Metaheuristics, C.C. Ribeiro, P. Hansen (Eds.), Kluwer, Norwell, MA, 2002, pp, 309-324.

[41] M.Desrochers, J. Desrosiers, M. Solomon, A new optimization algorithm for the vehicle routing problem with time windows, Operations Research. 40 (2) (1992) 342-354.

[42] J. Desrosiers, F. Soumis, M. Desrochers, Routing with time windows by column generation, Networks. 14 (1984) 545-565.

[43] O. du Merle, D. Villeneuve, J. Desrosiers, P. Hansen, Communication stabilized column generation, Discrete Mathematics. 194 (1999) 229-237.

[44] J. Edmonds, Minimum partition of a matroid into independent subsets, J. Res. Nat. Bur. Standards Sect. B. 69 (1965) 67-72.

[45] J. Edmonds, Maximum matching and a polyhedron with 0, l-vertices., J. Res. Nat. Bur.Standards Sect B. 169 (1965) 125 - 130.

[46] J. Edmonds, Paths, tree and flowers, Canadian Journal of Mathematics. 17 (1965) 449-467.

[47] S. Elhedhli, J.L. Goffin, The Integration of an interior-point cutting plane method within a Branch-and-price algorithm, Mathematical Programming.100 (2004) 267-294.

[48] L.F. Escudero, S. Muñoz, On identifying dominant cliques, European Journal Of Operations Research. 149 (2003) 65–76.

[49] I.R. de Farias, E.L. Johnson, G.L. Nemhauser, Branch-and-cut for combinatorial optimization without auxiliary 0-1 variables, Knowledge Eng Rev. 16 (2001) 25–39.

[50] L.R. Ford, Jr., D.R. Fulkerson, Flows in Networks, Princeton University Press, Princeton, 1962.

[51] A. Frank, Some polynomial algorithms for certain graphs and hypergraphs, in Proceedings of 5th British Combinatorial Conference, Winnipeg, Manitoba, Canada, 1975, pp. 211–226.

[52] A. Frangioni, Generalized bundle methods, SIAM Journal on Optimization. 13 (1) (2002) 117-156.

[53] A. Frangioni, G. Gallo, A bundle type dual-ascent approach to linear multicommodity min-cost flow problems, INFORMS Journal on Computing, 11 (4) (1999) 370-393.

[54] D.R. Fulkerson, Blocking polyhedra, in: Graph Theory and its Applications, B. Harris (Ed.), Academic Press, New York, 1970, pp, 93-112.

[55] D.R. Fulkerson, Blocking and antiblocking pairs of polyhedra, Mathematical Programming. 1 (1971) 168-194.

[56] D.R. Fulkerson, Anti-blocking polyhedra, Journal of Combinatorial Theory B. 12 (1972) 50-71.

[57] E.J. Gardiner, P.J. Artymiuk, P.Willett, Clique-detection algorithms for matching three-dimensional molecular structures, J Mol Graph Model. 15 (1998) 245–253.

[58] M. Garey, D. Johnson, Computers and Intractability,W.H. Freeman and Company, New York, 1979.

[59] P.C. Gilmore, R.E. Gomory, A linear programming approach to the cutting stock problem, Operations Research. 9 (1961) 849-859.

[60] P.C. Gilmore, R.E. Gomory A linear programming approach to the cutting stock problem- part II, Operations Research. 11 (6) (1963) 863-888.

[61] J.L. Goffin, A. Haurie, J.P. Vial, Decomposition and nondifferentiable optimization with the projective algorithm, Management Science. 38 (2) (1992) 284-302.

[62] J.L. Goffin, A. Haurie, J.P. Vial, D.L. Zhu, Theory and methodology: Using central prices in the decomposition of linear programs, European Journal of Operations Research. 64 (1993) 393-409.

[63] J.L. Goffin, J.P. Vial, Cutting planes and column generation techniques with the projective algorithm, Journal of Optimization Theory and Applications. 65 (3) (1990) 409-429.

[64] R.E. Gomory, Outline of an algorithm for integer solutions to linear programs, Bulletin of the American Mathematical Society.64 (1958) 275-278.

[65] R.E. Gomory, An Algorithm for the mixed integer problem, Technical Report RM-2597, The RAND Corporation, 1960.

[66] H.M. Grindley, P.J. Artymiuk, D.W. Rice, P.Willett, Identification of tertiary structure resemblance in proteins using a maximal common subgraph isomorphism algorithm, J Mol Biol. 229 (1993) 707–721.

[67] M. Grötschel, L. Lovász, A. Schrijver, Geometric Algorithms and Combinatorial Optimization, Springer-Verlag, Berlin, 1988.

[68] I.V. Hicks, Branch decompositions and minor containment, Networks. 43 (1), 2004 1-9.

[69] S. Holm, J. Tind, A unified approach for price directive decompostion procedures in integer programming, Discrete Applied Mathematics. 20 (1988) 205-219.

[70] R. Horaud, T. Skordas, Stereo correspondence through feature grouping and maximal cliques, IEEE Trans Pattern Anal Machine Intell. 11 (1989) 1168–1180.

[71] D. Huisman, R. Jans, M. Peeters, A.P.M. Wagelmans, Combining Column Generation and Lagrangian Relaxation, in: G. Desaulniers, J.Desrosiers, and M. M.Solomon (Eds), Column Generation, SpringerLink, US, 2006, pp. 247-270.

[72] K. Jansen, P. Scheffler, G. Woeginger, The disjoint cliques problem, Operations Research. 31 (1997) 45–66.

[73] D. Johnson, M. Trick (Editors), Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge, AMS, Providence, RI, 1996.

[74] R.G. Jeroslow, Cutting Plane Theory: Disjunctive Methods, Annals of Discrete Mathematics. 1 (1977) 293-330.

[75] G. Karypis, V. Kumar, A fast and high quality multilevel scheme for partitioning irregular graphs, SIAM J Sci Comput. 20 (1998) 359–392.

[76] G. Karypis, V. Kumar, Multilevel algorithms for multiconstraint graph partitioning, Technical Report 98-019, Army HPC Research Center, Department of Computer Science, University of Minnesota, Minneapolis, 1998.

[77] G. Karypis and V. Kumar, Multilevel k-way partitioning scheme for irregular graphs, J Parallel Distrib Comput. 48 (1998) 96–129.

[78] J.E. Kelley Jr., The Cutting-plane Method for solving convex programs, Journal for Society of Industrial Applied Mathematics. 8 (4) (1960) 703-712.

[79] S. Kim, K.N. Chang, J.-Y. Lee, A descent method with linear programming subproblems for nondifferentiable convex optimization, Mathematical Programming. 71. (1995) 17-28.

[80] K. Kim, J.L. Nazareth, The decomposition principle and algorithms for linear programming, Linear Algebra and Its Applications. 152 (1991) 119-133.

[81] L. Ladányi, T.K. Ralphs, L.E. Trotter Jr, Branch, cut, and price: Sequential and parallel, Computational Comb. Optimization LNCS. 2241 (2001) 223-260.

[82] C. Lemaréchal, Bundle methods in nonsmooth optimization, vol. 3 of IIASA Proceedings Series, C. Lemaréchal and R. Miffin (eds), Pergamon Press, Oxford, 1978.

[83] C. Lemaréchal, A. Nemirovskii, Y. Nesterov, New variants of bundle methods, Mathematical Programming, 69 (1995) 111-147.

[84] C. Lemaréchal, C. Sagastizabál, Variable metric bundle methods: From conception to implementation, Mathematical Programming. 76(3) (1997) 393-410.

[85] L.A.N. Lorena, E.L.F. Senne, A column generation approach to capacitated p-median problems, Computers and Operations Research. 31 (2004) 863-876.

[86] L. Lovasz, A. Schrijver, Cones of matrices and set functions and 0-1 optimization, SIAM Journal of Optimization. 1 (1991) 166-190.

[87] M.E. Lübbecke, J. Desrosiers, Selected topics in column generation, Optimization Online.
http://www.optimization-online.org/DB_HTML/2002/12/580.html (Last Modified 2005).

[88] J. MacWilliams, N.J.A. Sloane, The Theory of ErrorCcorrecting Codes, North-Holland, Amsterdam, 1979.

[89] C. Mannino, A. Sassano, An exact algorithm for the maximum stable set problem, Comput Optimizat Appl. 3 (1994), 243–258.

[90] R. E Marsten, W.W. Hogan, J.W. Blankenship, The boxstep method for large-scale optimization, Operations Research. 23 (3) (1975) 389-405.

[91] R.K. Martinson, J. Tind, An interior point method in Dantzig-Wolfe decomposition, Computers and Operations Research. 26 (12) (1996) 1195-1216..

[92] A. Mehrotra, M.A. Trick, A column generation approach for graph coloring, INFORMS J Comput. 8 (1996) 344–354.

[93] E.M. Mitchell, P.J. Artymiuk, D.W. Rice, P. Willet, Use of techniques derived from graph theory to compare secondary structure motifs in proteins, J Mol Biol. 212 (1989) 151–166.

[94] L. Nazareth, Numerical behavior of LP algorithms based upon the decomposition principle, Linear Algebra and Its Applications. 57, (1984) 181-189.

[95] G.L. Nemhauser, G. Sigismondi, A strong cutting plane/branch-and-bound algorithm for node packing, Journal Of Operational Research Society. 43 (1992) 443–457.

[96] G.L. Nemhauser, L.E. Trotter, Properties of vertex packings and independence system polyhedra, Mathematical Programming. 6 (1974) 48-61.

[97] G.L. Nemhauser, L.E. Trotter, Vertex packings: Structural properties and algorithms, Mathematical Programming. 8 (1975) 232-248.

[98] G.L. Nemhauser, L.A.Wolsey, Integer and Combinatorial Programming, John Wiley and Sons, New York, 1988.

[99] P.R.J. Ostergard, A fast algorithm for the maximum clique problem, Discrete Appl Math. 120 (2002) 197–207.

[100] M.Padberg, On the facial structure of the set packing polyhedra, Mathematical Programming. 5 (1973) 199-216.

[101] P.M. Pardalos, G.P. Rodgers, A branch and bound algorithm for the maximum clique problem, Comput Oper Res. 19 (1992), 363–375.

[102] G.R. Parija, R. Gadidov, W. E. Wilhelm, A facet generation procedure for solving 0/1 integer programs, Operations Research. 47:(5) (1999) 789-791.

[103] A. Pigatti, M. Poggi de Aragao, Stabilized Branch-and-cut-and-price for the generalized assignment problem, Working Paper, Departmento de Informatica, PUC do Rio de Janeiro, Brazil (2004).

[104] T.K. Ralphs, M. V. Galati Decomposition and dynamic cut generation in integer programming, Optimization Online.
http://www.optimization-online.org/DB_HTML/2003/09/726.html (Last Modified 2005).

[105] T.K. Ralphs, and L. Ladanyi SYMPHONY: A parallel framework for branch, cut and price, White Paper, Rice University, (2000) 1-19.

[106] T.K. Ralphs, L. Ladanyi, M.J. Saltzman, Parallel branch, cut, and price for large-scale discrete optimization, Mathematical Programming. 98 (2003) 253-280.

[107] R.T. Rockafellar, Monotone operators and the proximal point algorithm, SIAM Journal of Control and Optimization. 14 (5) (1976) 877-898.

[108] R.T. Rockafellar, Convex Analysis, Princeton University Press, New Jersey, 1970.

[109] F. Rossi, S. Smriglio, A branch-and-cut algorithm for the maximum cardinality stable set problem, Operations Research Letters. 28 (2001) 63–74.

[110] L.M. Rousseau, M. Gendreau, D. Feillet, Interior point stabilization for column generation, C.R.T. publication C7PQMR PO2003-39-X, 1-10, Operations Research Letters. (In Press, Corrected Proof December 2006).

[111] A. Schrijver, Theory of Linear and Integer Programming, John Wiley & Sons, New York, 1986.

[112] E.L.F. Senne, L.A.N. Lorena, Stabilizing column generation using Lagrangean/surrogate relaxation: An application to P-median location problems, EURO 2001 – the European Operational Research Conference, Erasmus University, Rotterdam, July 2001, 1-23.

[113] E.C. Sewell, A branch and bound algorithm for the stability number of a sparse graph, INFORMS Journal of Computing. 10 (1998) 438–447.

[114] H. Sherali, W. Adams, A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems, SIAM Journal on Discrete Mathematics. 3 (1990) 411-430.

[115] H. Sherali, C. Shetty, Optimization with disjunctive constraints, Lecture Notes in Economics and Mathematical Systems 181, Springer, 1980.

[116] L.E. Trotter Jr., A class of facet producing graphs for vertex packing polyhedra. Discrete Math. 12 (1975) 373–388.

[117] L.E. Trotter Jr., Solution characteristics and algorithms for the vertex packing problem. Technical Report 168, Dept. of Operations Research, Cornell University, Ithaca, NY, 1973.

[118] M.Van den Akker, H. Hoogenveen, S. van de Velde (2002) Combining column generation and Lagrangean relaxation to solve a single-machine common due date problem, INFORMS Journal on Computing. 14:(1) (2002) 37-51.

[119] F. Vanderbeck, Automated Dantzig-Wolfe reformulation or how to exploit simultaneously original formulation and column generation re-formulation, Working paper, Department of Applied Mathematics, University of Bordeaux, Talence Cedex, France, 2003.

[120] F. Vanderbeck, Computational study of a column generation algorithm for bin packing and cutting stock problems, Math. Programming Series A. 86 (1999) 565-594.

[121] F. Vanderbeck, On Dantzig-Wolfe Decomposition in Integer Programming and Ways to Perform Branching in a Branch-and-Price Algorithm, Operations Research. 48 (1) (2000) 111-128.

[122] F. Vanderbeck, A Generic View at the Dantzig-Wolfe decomposition approach in mixed integer programming: Paving the way for a generic code, Presentation, Laboratoire de Mathematique Appliques de Bordeaux, University of Bordeaux, Talence Cedex, France, 2004.

[123] A.M.Verweij, Selected applications of integer programming: A computational study, Ph.D. Thesis, University of Utrecht, Utrecht, Holland, September, 2000.

[124] B. Verweij, K. Aardal, An optimization algorithm for maximum independent set with applications in map labeling, Proceedings of the Seventh Annual European Symposium on Algorithms, Number 1643, J. Nesetril (Editor), in Lecture Notes in Computer Science, Springer-Verlag, Berlin, 1999, pp. 426–437.

[125] D. Villeneuve, J. Desrosiers, M. E. Lubbecke, F. Soumis, On compact formulations for integer programs solved by column generation, Annals of Operations Research. 139:(1) (2005) 375-388.

[126] S. de Vries, R. Vohra, Combinatorial auctions: A survey, INFORMS J Comput. 15 (2003) 284–309.

[127] D. Warrier, W.E. Wilhelm, J. Warren, I.V. Hicks, A branch-and-price approach for the maximum weighted independent set problem, Networks. 24 (2005) 198–209.

[128] D. West, Introduction to Graph Theory, Prentice-Hall, Upper Saddle River, NJ, 2001.

[129] P.Wentges, Weighted Dantzig-Wolfe decomposition of linear mixed-integer programming, International Transactions Operational Research. 4:(2) (1997) 151-162.

[130] W.E. Wilhelm, A technical review of column generation in integer programming, Optimization Engineering. 2 (2001) 159–200.

[131] W.E. Wilhelm, S.Sachdeva, Vertex cloning to facilitate a branch-and-price approach to the maximum weighted independent set problem, (Working Paper) Texas A&M University, College Station, December 2006.

[132] J. Xue, Edge-maximal triangulated subgraphs and heuristics for the maximum clique problem, Networks. 24 (1994) 109–120.

APPENDIX A

A GENERIC IMPLEMENTATION OF BRANCH AND PRICE

Here, we describe the generic implementation of B&P used in our research. The goal of this implementation is to provide a reusable framework, which can be easily adapted for different applications of B&P. In addition to the standard techniques for B&P, our implementation provides advanced techniques for branching, stabilizing DWD and generating cuts. Moreover, the implementation offers flexibility for invoking future enhancements. Our implementation is in C++ with embedded CPLEX Callable Library routines. The implementation comprises the following components:

B&P generic routines: This set of routines provides the standard implementation of the B&P algorithm. We implement DWD column generation using CPLEX callable library routines at each node of the B&B tree. We initialize RMP using artificial variables. Both two-phase and Big-M methods are implemented for initializing RMP. Improving columns generated are preserved in a column pool for future iterations. We provide to the user the option of entering a single improving column or all improving columns into RMP. We also provide an option for identifying an improving column from the column pool before solving the pricing subproblems. The B&B tree is searched according to a breadth-first strategy. Our implementation allows the user to invoke both variable dichotomy branching as well as constraint branching. Parent-node columns that are feasible with respect to a child node are used to initialize RMP for the child node.

Problem specific routines: This set of routines is used to invoke problem-specific information and interface with the generic routines discussed above. The user is required

to populate data structures to define the specific instance at hand. The user is also required to specify an oracle to solve the pricing subproblems.

B&P stabilization routines: This set of routines provides additional enhancements for improving the rate of convergence of DWD. Prevalent stabilization methods are provided. Specifically, the Boxstep method, 3-piece and 5-piece penalty function method, and Wentges smoothing method are implemented. The user is required to specify the associated parameters for each of these methods.

B&P cut generation routines: This set of routines invokes cut generation strategies at the root node of the B&P tree. Specifically, the two strategies developed in this dissertation are implemented.

VITA

Name: Deepak Warrier

Address: 501 Sycamore Lane #1121
EulessTX-76039

Email Address: warrier@neo.tamu.edu

Education: B.Tech., Mechanical Engineering R.E.C., Calicut, INDIA 1998
M.S., Industrial Engineering Texas A&M University, 2002
Ph.D., Industrial Engineering Texas A&M University, 2007