DEVELOPMENT OF A BRANCH AND PRICE APPROACH INVOLVING VERTEX CLONING TO SOLVE THE MAXIMUM WEIGHTED INDEPENDENT SET PROBLEM

A Thesis

by

SANDEEP SACHDEVA

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

MASTER OF SCIENCE

December 2004

Major Subject: Industrial Engineering

DEVELOPMENT OF A BRANCH AND PRICE APPROACH INVOLVING VERTEX CLONING TO SOLVE THE MAXIMUM WEIGHTED INDEPENDENT SET PROBLEM

A Thesis

by

SANDEEP SACHDEVA

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

MASTER OF SCIENCE

Approved as to style and content by:

December 2004

Major Subject: Industrial Engineering

ABSTRACT

Development of a Branch and Price Approach Involving Vertex Cloning to Solve the Maximum Weighted Independent Set Problem. (December 2004)

Sandeep Sachdeva, B.Tech., Indian Institute of Technology, Delhi, India

Chair of Advisory Committee: Dr. Wilbert E. Wilhelm

We propose a novel branch-and-price (B&P) approach to solve the maximum weighted independent set problem (MWISP). Our approach uses clones of vertices to create edge-disjoint partitions from vertex-disjoint partitions. We solve the MWISP on sub-problems based on these edge-disjoint partitions using a B&P framework, which coordinates sub-problem solutions by involving an equivalence relationship between a vertex and each of its clones. We present test results for standard instances and randomly generated graphs for comparison. We show analytically and computationally that our approach gives tight bounds and it solves both dense and sparse graphs quite quickly.

DEDICATION

To my parents

ACKNOWLEDGEMENTS

I would like to sincerely thank my advisor Dr. Wilbert E. Wilhelm for providing motivation and guidance in pursuing this research. In addition, I would like to acknowledge the support of Dr. Illya Hicks and other members of the NSF project team for their valuable suggestions. I am also thankful to my friends for their support throughout my stay at Texas A&M University.

This material is based in part upon work supported by the Texas Advanced Technology Program on Grant Number 000512-0248-2001 and by the National Science Foundation on Grant DMI-9500211.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER I

INTRODUCTION

1.1 Overview

Given a graph, $G = (V, E)$ where V represents the set of vertices; and E, the set of edges, a subset of vertices $I \subseteq V$ such that no two vertices in I are adjacent to each other constitutes an independent set (IS). The problem of finding the independent set of largest cardinality in a graph is known as the maximum independent set problem (MISP). The cardinality of the maximum independent set is known as the independence number or the stability number of the graph. Extending the MISP to vertex-weighted graphs, the MWISP is to find the independent set of maximum weight. Letting w_v represent the weight associated with vertex v for $v \in V$, the MWISP is to find the independent set I such that $\sum_{v \in I} w_v$ is maximized. Both MISP and MWISP are known to be NP-Hard [12]. Even though the MWISP can be solved in polynomial time on some specialized graph structures ([1], [11]); the problem remains NP-Hard on arbitrary graphs.

MISP and MWISP are among the most researched problems in the field of graph theory. They have large numbers of practical applications in diverse fields, including protein structure realignment [8], coding theory [7], computer vision [2], experimental design [2], signal transmission [2], and information retrieval [2].

This thesis follows the style and format of the European Journal of Operational Research.

1.2 Motivation and Objectives

The approach explored in this study involves solving the integer programming formulation of the MWISP (in edge inequality form) which may be stated as

$$Z^*_{MWISP} = Max\left\{ \sum_{v \in V} w_v x_v : \mathbf{x} \in Q \right\}, \text{ where } \quad Q = \left\{ \mathbf{x} \in B_+^{|V|} : x_u + x_v \le 1 \ \forall (u,v) \in E \right\}, \qquad (1)$$

where $x_v = 1$ if vertex v is included in the independent set, and $x_v = 0$ otherwise.

Warrier et al [25] developed a branch-and-price (B&P) approach to solve the MWISP and showed that their approach gives competitive results for sparse graphs. However, their approach suffers from two major drawbacks: their restricted master problem (RMP) gives bounds that are not tight and comprises a large number of constraints, requiring lengthy run times. This study contributes a new B&P approach, which is directed towards overcoming these shortcomings. This new approach, which we call *Vertex Cloning*, is designed to facilitate solution by yielding a RMP with fewer constraints. We also show that Vertex Cloning provides a tighter formulation, improving bounds in the branch-and-bound (B&B) tree.

The primary objectives of this study are:

(1) Formulation of the Vertex Cloning approach,

(2) Analysis showing that Vertex Cloning yields a tighter formulation,

(3) Effective methods to implement Vertex Cloning, and

(4) Analysis of the computational efficacy of Vertex Cloning.

1.3 Basic Notations

We consider only simple, undirected, and finite graphs. Most of the notation we use in this thesis is the same as that used by Warrier et al [25]. We represent an edge as $e \in E$ or, alternatively, by denoting its end vertices as $(u,v) \in E$ where $u,v \in V, u \neq v$. We use $G' = (V', E')$ to denote the complement graph of G, where $V' = V$ and $E' = \{(u,v) \notin E : u,v \in V, u \neq v\}$. We use $N(v)$ to denote the set of v's neighbors, $\{u : (u,v) \in E\}$.

We decompose graph G into $|P|$ sets of vertex-induced partitions. We use $G_p = (V_p, E_p)$ for $p \in P$ to denote the sub-graph (partition) p, where V_p and E_p denote the set of vertices and edges in partition p, respectively. Furthermore, we use $\hat{E}$ to represent the set of edges that connect vertices in different sub-graphs, $\hat{E} = E \setminus \bigcup_{p \in P} E_p$; and similarly, $\hat{V}$ to denote the set of vertices at the ends of edges in $\hat{E}$. For $v \in V$, we use π_v to identify the partition into which v is assigned. We use $N_p(v)$ to denote the neighbors of v in partition $p \in P$. Vertex Cloning may duplicate certain vertices into partition $p \in P$. We use an "over bar" to denote the vertex and edge sets in partition p after duplication (i.e., $\overline{V}_p$ and $\overline{E}_p$ for $p \in P$).

1.4 Organization of Thesis

The remainder of this thesis is organized in six chapters. Chapter II presents a review of the literature on MWISP, including a detailed discussion of the B&P approach developed by Warrier et al [25]. Chapter III introduces concepts that underlie Vertex Cloning and gives a detailed mathematical formulation (objective 1). Chapter IV discusses properties of polyhedra formed by various B&P formulations (objective 2). Chapter V discusses implementation issues (objective 3) and Chapter VI analyzes computational results (objective 4), comparing the performance of several algorithms for solving the MWISP. Finally, Chapter VII gives summary and recommendations for future research.

CHAPTER II

LITERATURE REVIEW

A solution to the MWISP can be obtained as the solution to the maximum weighted clique problem on the complementary graph and the literature describes extensive study of both problems. The solution methods presented in the literature use variety of approaches for solving the MWISP, which includes B&B [2, 3, 7, 21], implicit enumeration [9] and standard heuristic methods like genetic algorithms [13] and greedy random adaptive search procedures [10]. Bomze et al [6] gave an extensive survey of algorithms, complexity and applications of maximum clique problem. Recently, Carr et al [8] described a branch-and-cut approach for the MWISP.

Bazaara et al [5] gave a good description of Dantzig-Wolfe decomposition (DWD) for linear programming problems. DWD may be applied to the linear relaxation of an integer programming problem to obtain a bound at each node in the B&B tree in an approach known as B&P. Over the last twenty years, B&P has been successfully applied in a wide range of integer programming problems [4, 18, 20, 23, 24, 26]. To apply B&P, integer programming problems must be decomposed into two sets of constraints; those that form sub-problem(s) and those that are relegated to the RMP. Barnhart et al [4] and Wilhelm [26] provided extensive overviews of B&P and gave descriptions of decomposition methods, and associated implementation issues.

Mehrotra and Trick [18] used B&P to solve the minimum coloring problem, another important graph problem. The minimum coloring problem is to find the minimum number of colors that allows each vertex to be colored so that the endpoints of each edge have different colors. They used a set covering formulation of the coloring problem with the objective of finding the minimum number of maximal independent sets such that the union of these sets includes all vertices of the graph. Their RMP consisted of set covering constraints and their (single) sub-problem involved finding the maximal independent set.

Warrier et al's [25] B&P approach partitions a graph into smaller, vertex-disjoint sub-graphs and solves a MWISP on each sub-graph (sub-problem) to generate columns that are coordinated by a RMP to obtain the MWIS for the original graph. Their approach partitions the inequalities associated with edge constraints in (1) into two sets; one set, the coordinating set, comprises inequalities associated with edges that connect vertices in different partitions (i.e., $x_u + x_v \leq 1 \; \forall (u,v) \in \hat{E}$); and the other set, $|P|$ sub-problems, each consisting of inequalities associated with the respective edges included in a partition (i.e., $x_u + x_v \leq 1 \; \forall (u,v) \in E_p$) . They used B&P, forming the RMP (we duplicate their model here) as:

$$Z^*_{LP} = Max \sum_{p=1}^{P} \sum_{j \in J_p} \lambda_{jp} (\mathbf{w}^p \mathbf{x}^{jp}) \tag{2}$$

$$\text{s.t.} \qquad \sum_{p=1}^{P} \sum_{j \in J_p} \lambda_{jp} (A_p \mathbf{x}^{jp}) \leq 1 \tag{3}$$

$$\sum_{j \in J_p} \lambda_{jp} = 1 \qquad \forall \; p \in P \tag{4}$$

$$\lambda_{jp} \geq 0 \qquad \forall \; p \in P, \; j \in J_p \tag{5}$$

where

J_p denotes the set of integer extreme points of $conv(Q_p \cap B^{|V_p|})$,

$\mathbf{x}^{jp}$ is a $|V_p|$- vector that defines extreme point $j \in J_p$, and

λ_{jp} is a RMP decision variable that corresponds to extreme point $j \in J_p$.

Sub-problem $p \in P$ is formulated as

$$Z_p^* = Max\left\{(w^p - A_p^T \alpha)\mathbf{x}^{jp} : \mathbf{x}^{jp} \in Q_p \cap B^{|V_p|}\right\}, \tag{6}$$

where $Q_p = \left\{\mathbf{x}^p \in R_+^{|V_p|} : x_u + x_v \le 1 \ \forall (u,v) \in E_p\right\}$ and α is an $|\hat{E}|$-vector of dual variables associated with constraint (3).

They tested two different partitioning procedures; one partitioned an original graph into chordal sub-graphs and the other used METIS [15, 16, 17], a heuristic that seeks to minimize the number of edges in $\hat{E}$, while balancing the number of vertices in different partitions, given the number of partitions. They solved MWISP on each chordal sub-graph using Frank's algorithm [11]. For solving the NP-Hard MWISP posed by each METIS-partitioned sub-graph, they modified the Carraghan and Pardalos [9] algorithm to address weights and solve the MWISP in the graph (the original algorithm finds the maximal clique in a graph). We refer to this modified algorithm using the acronym MCP. In addition to evaluating these two methods to partition a graph, they tested with two types of RMP formulation and two methods of branching. They tested their methodology with DIMACS Challenge Problems [14] and randomly generated p-graphs and concluded that the combination of METIS partitioning, RMP formulation in terms of clique inequalities and branching on cliques in B&B tree gave the best results.

Furthermore, they found that their method outperformed the MCP algorithm for sparse graphs, which are known to be especially challenging. Subsequently, we refer to this as the Original B&P (OBP) approach to solve the MWISP.

CHAPTER III

VERTEX CLONING APPROACH

This chapter introduces Vertex Cloning (henceforth referred to as Cloning) and its mathematical formulation.

3.1 Concept

Cloning extends the partitioning methods employed by Warrier et al [25] by cloning selected vertices with the goal of eliminating edges in set $\hat{E}$. After using METIS to partition the graph $G=(V,E)$ into $|P|$ disjoint sub-graphs $G_1,\ldots\ldots,G_{|P|}$, each edge $e=(u,v)\in\hat{E}$ connects vertices in two different partitions ($u\in V_p, v\in V_q$ where $p,q\in P, p\neq q$) and the associated edge inequality ($x_u+x_v\leq 1$) is included in the RMP. Cloning can duplicate vertex u (v) into partition q (p) so that edge (u,v) lies entirely in partition q (p) and the edge inequality in the RMP can be replaced by an equality $x_w=x_u$ (x_v), where w is the clone of u (v). Similarly, edge inequalities in the RMP can be replaced by relationships equating the decision variables associated with a cloned vertex and each of its clones.

Cloning is analogous to the cost splitting technique of Lagrange relaxation [19, 22] through which, depending on the structure of the problem, duplicate variables can be introduced to improve bounds. We refer to a vertex that is duplicated as the *cloned* *(originating)* vertex and any duplicate vertex as a c*lone*. We use the term *copies* to indicate an original vertex along with its clones.

We illustrate Cloning using Figure 1, which depicts a graph comprising 7 vertices and 7 edges. The formulation for the MWISP on this graph (as in (1)) can be written as:

$$Max\, Z_{IP} = x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 \tag{7}$$

s.t.

$$x_1 + x_2 \le 1 \tag{8}$$

$$x_1 + x_5 \le 1 \tag{9}$$

$$x_2 + x_3 \le 1 \quad 10)$$

$$x_2 + x_7 \le 1 \tag{11}$$

$$x_3 + x_4 \le 1 \tag{12}$$

$$x_3 + x_6 \le 1 \tag{13}$$

$$x_4 + x_5 \le 1 \tag{14}$$

$$\mathbf{x} = (x_1, x_2, x_3, x_4, x_5, x_6, x_7) \in Z_+^7 \tag{15}$$

Figure 2 shows an arbitrary partitioning with $P = \{1,2\}$, where $V_1 = \{v_1, v_2, v_6, v_7\}$ and $V_2 = \{v_3, v_4, v_5\}$. Let G_1 and G_2 represent the two sub-graphs (partitions), respectively, and $\hat{E} = \{(v_1, v_5), (v_3, v_6), (v_2, v_3)\}$ be the set of edges that connects vertices in the partitions. The endpoints of all edges $e \in \hat{E}$ comprise the set $\hat{V} = \{v_1, v_2, v_3, v_5, v_6\}$.

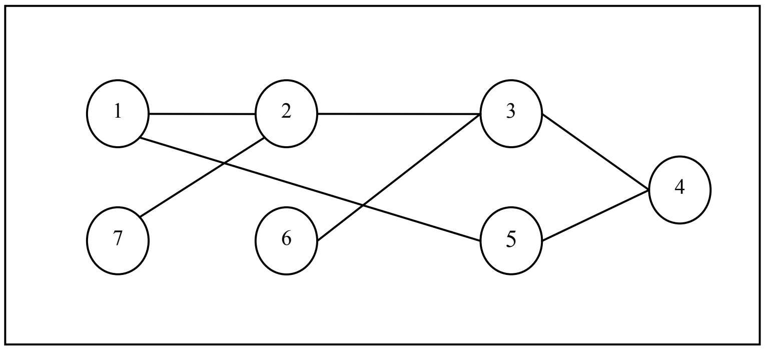

Fig. 1 Example graph *G*.

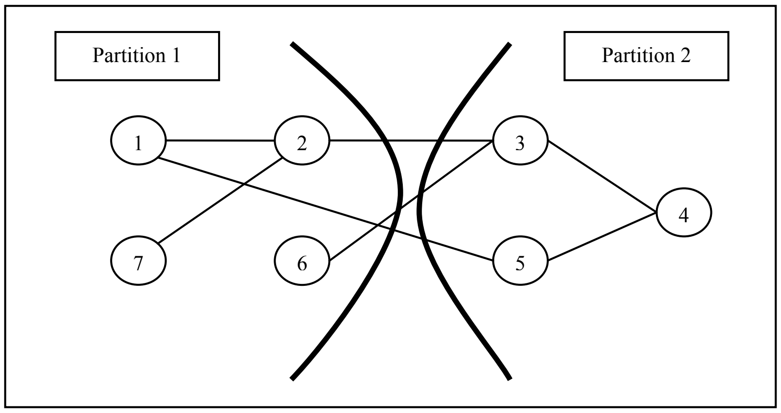

Fig. 2 Vertex disjoint partitions of *G*.

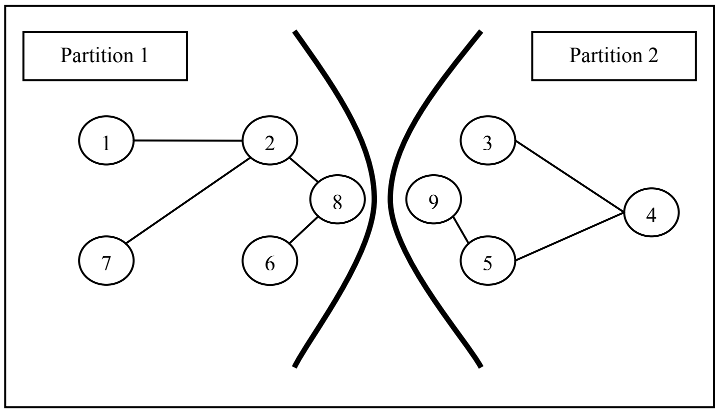

Fig. 3 Edge disjoint partitioning of *G* through vertex cloning.

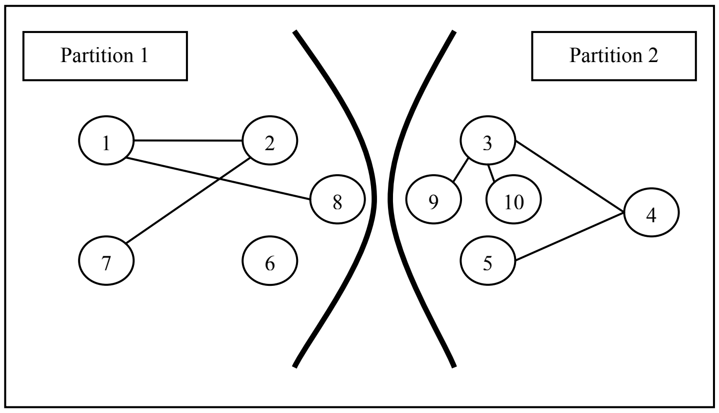

Fig. 4 Edge disjoint partitioning of *G* by cloning different vertices.

The OBP reformulates model (7)-(15), creating one sub-problem with edge inequalities associated with G_1 (i.e., (8) and (11)) and an other sub-problem with edge inequalities associated with G_2 (i.e., (12) and (14)). The RMP comprises inequalities corresponding to edges $e \in \hat{E}$ (i.e., (9), (10) and (13)).

Figure 3 depicts one possible way to clone vertices so that all edge inequalities in the RMP are replaced with equality constraints. Here, v_3 is duplicated (as v_8) in partition 1 so that the edges (v_2, v_3) and (v_3, v_6) can be included in partition 1 as (v_2, v_8) and (v_6, v_8), respectively. Similarly, v_1 is cloned as v_9 in partition 2 to include edge (v_1, v_5) in partition 2 as (v_5, v_9). This cloning process results in an edge-disjoint partitioning of G in which $\hat{E} = \emptyset$ and equalities $x_{v_3} = x_{v_8}$ and $x_{v_1} = x_{v_9}$ replace corresponding edge inequalities ((9), (10) and (13)) to assure that decision variables associated with a cloned vertex and each of its clones are equal. Cloning results in vertices, instead of edges, being shared between partitions. Figure 4 demonstrates an alternate way to clone vertices. In this case, three clones (namely v_8, v_9 and v_{10}) are formed (as clones of v_5, v_6 and v_2, respectively). This alternate cloning adds more vertices into the partitions, making the sub-problems more challenging to solve and also resulting in a larger RMP. Thus, the approach should clone a minimum number of vertices to promote tractability.

Note that, typically, only a subset of vertices in $\hat{V}$ need be cloned to locate each edge $e \in \hat{E}$ into some partition. In Figure 3, only two vertices from the set $\hat{V}$ are cloned and in Figure 4, three vertices are cloned.

3.2 Formulation

We now specialize the MWISP to represent Cloning. Let $K \in \hat{V}$ be the set of cloned vertices and D_v denote the set of clones corresponding to vertex $v \in K \in \hat{V}$. Cloning vertex v (as w) relocates a set of edges $(u,v) \in \hat{E}$ from $\hat{E}$ to partition π_u (also, $\pi_w = \pi_u$). In partition π_u, this relocated edge(s) (u,v) exists as (u,w).

Note that not all vertices in $\hat{V}$ need be cloned (see example in 3.1). If vertex $v \in \hat{V}$ is not cloned, $D_v = \varnothing$ and if it is cloned, D_v gives the set of its clones. Let $K \in \hat{V}$ denote the set of vertices for which $D_v \neq \varnothing$. Cloning increases the number of vertices in the graph to $\left|\overline{V}\right|$, where $\overline{V} = V \bigcup_{v \in K} D_v$.

Cloning adds vertices and edges to certain partitions, changing $G_p = \left(V_p, E_p\right)$ to $\overline{G}_p = \left(\overline{V}_p, \overline{E}_p\right)$, where $\overline{V}_p$ includes V_p and clones that are added in partition p and $\overline{E}_p$ includes edges from set E_p as well as relocated edges. Correspondingly, the vector $\mathbf{x^p}$ is changed to $\overline{\mathbf{x}}^{\mathbf{p}} = \left\{x_v : v \in \overline{V}_p\right\}$. The integer programming formulation of the MWISP may now be specialized to reflect Cloning:

$$Z^{D}_{MWISP} = Max\left\{\sum_{p \in P} \sum_{v \in \overline{V}_p} w_v x_v : x_w - x_v = 0 \; \forall \; v \in K, \; w \in D_v \right\} \tag{16}$$

where $\overline{\mathbf{x}}^{\mathbf{p}} \in \overline{Q}^{\,p} \; \forall \, p \in P$ and $\overline{Q}^{\,p} = \left\{ \overline{\mathbf{x}}^{\mathbf{p}} \in B_{+}^{\left|\overline{V}_p\right|} : x_u + x_v \leq 1 \; \forall \, (u,v) \in \overline{E}_p \right\}$. The formulation given in (16) can be rewritten as follows:

$$Z_{MWISP}^{D} = Max \sum_{p \in P} \mathbf{w^p} \overline{\mathbf{x}}^{\mathbf{p}} \tag{17}$$

s.t.

$$\sum_{p \in P} A_p \overline{\mathbf{x}}^{\mathbf{p}} \qquad = 0 \tag{18}$$

$$B_p \overline{\mathbf{x}}^{\mathbf{p}} \qquad \leq 1 \qquad \forall\, p \in P \tag{19}$$

$$\overline{\mathbf{x}}^{\mathbf{p}} \qquad \in B^{\left|\overline{V}_p\right|} \quad \forall\, p \in P \tag{20}$$

where A_p denotes the matrix of coefficients of decision variables in equalities (18) and B_p denotes the matrix of coefficients of decision variables in inequalities (19).

Equalities (18) include an equivalence relation between each cloned vertex and each of its clones; and inequalities (19) include edge inequalities in partition $p \in P$. Inequalities (19) define $|P|$ disjoint blocks of constraints, one for each partition p, forming a block diagonal structure. Application of DWD to the linear relaxation of (17)-(20) allows each block to be addressed as an independent sub-problem while relegating constraint (18) to the RMP:

$$Z_{RMP}^{*} = Max \sum_{p \in P} \sum_{j \in J_p} \overline{\lambda}_{jp} (\mathbf{w}^{p} \overline{\mathbf{x}}^{jp}) \tag{21}$$

s.t.

$$\sum_{p \in P} \sum_{j \in J_p} \overline{\lambda}_{jp} (A_p \overline{\mathbf{x}}^{jp}) \qquad = 0 \tag{22}$$

$$\sum_{j \in J_p} \overline{\lambda}_{jp} \qquad = 1 \qquad \forall\, p \in P \tag{23}$$

$$\overline{\lambda}_{jp} \qquad \geq 0 \qquad \forall\, p \in P,\ j \in \overline{J}_p \tag{24}$$

where

$\overline{J}_p$ denotes the set of integer extreme points of $conv(\overline{Q}_p)$,

$\overline{\mathbf{x}}^{jp}$ is a $|\overline{V}_p|$- vector that defines extreme point $j \in J_p$ and

$\overline{\lambda}_{jp}$ denotes the RMP decision variable that corresponds to extreme point $j \in J_p$.

Sub-Problem $p \in P$ is a MWISP of the form:

$$Z_p^* = Max\ \hat{c}_j - \hat{z}_j = Max\left\{(\mathbf{w}^p - \alpha A_p)\overline{\mathbf{x}}^{jp} - \beta_p : \mathbf{x} \in \overline{Q}^p\right\}, \tag{25}$$

where α is a vector of dual variables associated with equality constraints (22) and β_p is the dual variable associated with convexity constraint p in (23).

Optimal extreme point j in sub-problem p gives vector $\overline{\mathbf{x}}^{jp}$, which is an improving column if $Z_p^* > 0$. At each iteration, we solve all $|P|$ sub-problems and select $\overline{\mathbf{x}}^{jp}$ as $\arg\max_{p \in P}\left(Z_p^*\right)$ to enter the RMP basis. If $Z_p^* \leq 0$ for all $p \in P$, the current RMP solution is optimal.

CHAPTER IV

ANALYSIS OF BOUNDS

In this chapter we analyze the polytope associated with the OBP (given in (2)-(6)) and Cloning (given in (21)-(25)) models and their linear relaxations to show that Cloning gives a tighter bound at the root node of B&B tree than that obtained by OBP. Our proof is based on showing that the polytope associated with Cloning is contained in the polytope associated with the OBP. To promote simplicity, we present our discussion in terms of the polytopes associated with decision variables x_v.

Let S denote the set of feasible integral solutions to (1); C, the convex hull of S; and L, the polytope associated with the linear relaxation of (1):

$$S = \left\{ \mathbf{x} \in Z_+^{|V|} : x_u + x_v \leq 1 \ \forall \ (u,v) \in E, \ x_v \in \{0,1\} \ \forall v \in V \right\},$$

$$C = conv(S) \text{ and}$$

$$L = \left\{ \mathbf{x} \in R_+^{|V|} : x_u + x_v \leq 1 \ \forall \ (u,v) \in E, \ 0 \leq x_v \leq 1 \ \forall v \in V \right\}.$$

Relative to the vertex-disjoint partitions formed in the OBP (see Chapter II), let S_{SP_p} and S_{CS} denote the set of integral solutions that are feasible relative to the edge inequalities in E_p (which constitute *block-diagonal* set $p \in P$) and $\hat{E}$ (which constitute the *coordinating set*), respectively:

$$S_{SP_p} = \left\{ \mathbf{x} \in Z_+^{|V|} : x_u + x_v \leq 1 \ \forall \ (u,v) \in E_p, \ x_v \in \{0,1\} \ \forall v \in V \right\} \text{ and}$$

$$S_{CS} = \left\{ \mathbf{x} \in Z_+^{|V|} : x_u + x_v \leq 1 \ \forall \ (u,v) \in \hat{E}, \ x_v \in \{0,1\} \ \forall v \in V \right\}.$$

Similarly, let $C_{SP_p} = conv(S_{SP_p})$ and $C_{CS} = conv(S_{CS})$. Let L_{SP_p} denote the polytope corresponding to the linear relaxation of S_{SP_p} for $p \in P$; and L_{CS}, the polytope associated with the linear relaxation of S_{CS}. Following their respective definitions, we have $S \subseteq C \subseteq L$, $S_{SP_p} \subseteq C_{SP_p} \subseteq L_{SP_p}$ and $S_{CS} \subseteq C_{CS} \subseteq L_{CS}$.

Noting that $E = \hat{E} \bigcup_{p \in P} E_p$; and E defines S, C and L; E_p defines S_{SP_p}, C_{SP_p} and L_{SP_p}; $\hat{E}$ defines S_{CS}, C_{CS} and L_{CS}; we have

$$S = S_{CS} \bigcap_{p \in P} S_{SP_p} \;,\; C \subseteq C_{CS} \bigcap_{p \in P} C_{SP_p} \;, \text{and } L = L_{CS} \bigcap_{p \in P} L_{SP_p} \;. \tag{26}$$

Define polytope R_O by substituting (tightening) L, replacing L_{SP_p} with C_{SP_p}:

$$R_O = L_{CS} \bigcap_{p \in P} C_{SP_p} \;. \tag{27}$$

Since $S_{SP_p} \subseteq C_{SP_p} \subseteq L_{SP_p}$ and $S_{CS} \subseteq C_{CS} \subseteq L_{CS}$, we may write,

$$S_{CS} \bigcap_{p \in P} S_{SP_p} \subseteq C_{CS} \bigcap_{p \in P} C_{SP_p} \subseteq L_{CS} \bigcap_{p \in P} C_{SP_p} \subseteq L_{CS} \bigcap_{p \in P} L_{SP_p} \;,$$

$$S \subseteq C \subseteq R_O \subseteq L \,. \tag{28}$$

Cloning replaces every edge inequality $x_u + x_v \le 1$ (where $(u,v) \in \hat{E}$) in the coordinating set (of OBP) by an equality $x_v = x_w$ (vertex v in partition π_v is cloned as w into partition π_u) and an inequality corresponding to a clone, $x_u + x_w \le 1$ (associated with edge (u,w) in partition π_u). Let L'_{CS} denote the polytope that is formed by replacing all edge inequalities ($x_u + x_v \le 1$) in L_{CS} with equalities ($x_v = x_w$) and edge inequalities ($x_u + x_w \le 1$). L'_{CS} can be written as intersection of polytopes $L^{=}_{CS}$ and $L^{\le}_{CS}$, where $L^{=}_{CS}$ denotes the polytope associated with the equality constraints that result from cloning ($x_v = x_w$) and $L^{\le}_{CS}$ denotes the polytope comprising edge inequalities ($x_u + x_w \le 1$), each of which includes a decision variable associated with a clone:

$$\overline{L}^{=}_{CS} = \left\{ \overline{\mathbf{x}} \in R^{|\overline{V}|}_{+} : x_w = x_v \ \forall v \in K, w \in D_v, 0 \le x_v \le 1, v \in \overline{V} \right\},$$

$$\overline{L}^{\le}_{CS} = \left\{ \overline{\mathbf{x}} \in R^{|\overline{V}|}_{+} : x_w + x_u \le 1 \ \forall \ (u,v) \in \hat{E}, v \in K, w \in D_v : \pi_u = \pi_w, 0 \le x_v \le 1, v \in \overline{V} \right\} \text{ and}$$

$$L'_{CS} = L^{=}_{CS} \cap L^{\le}_{CS}.$$

Note that Cloning increases the number of decision variables to $|\overline{V}|$ so the polytopes L'_{CS}, $L^{=}_{CS}$ and $L^{\le}_{CS}$ are defined in $|\overline{V}|$-dimensional space. We now prove that L_{CS} and L'_{CS} are equivalent; (i.e., the set of solutions that are feasible with respect to L'_{CS} in terms of the decision variables that correspond to the original vertices, $x_v : v \in V$, is same as those associated with L_{CS}). We represent this equivalence by "$\equiv$".

Proposition 1: $L_{CS} \equiv L'_{CS}$.

Proof: Let $X = \{x_v : v \in V\}$ be any vector in L_{CS} and construct $\overline{X} = \{\overline{x}_v : v \in \overline{V}\}$, comprising a $|V|$- sub-vector of variables $\overline{x}_v$ associated with original vertices (which includes all vertices but clones) and a $|\overline{V} \setminus V|$- sub-vector associated with clones. In particular, for original vertices $v \in V$, set $\overline{x}_v = x_v$. For each vertex $v \in K \subseteq \hat{V} \subseteq V$, identify each of its clones, $w \in D_v \in \overline{V} \setminus V$ and set $\overline{x}_w = x_v$. From the construction, it is clear that $\overline{X}$ is feasible with respect to L'_{CS}.

It is important to note that L'_{CS} contains $\sum_{v\in K}|D_v|$ more variables (associated with clones) than L_{CS}, tending to increase the dimension of polyhedron L_{CS} by $\sum_{v\in K}|D_v|$. For $v\in K$, one equality constraint relates cloned vertex v to each of its clones $w(\bar{x}_w = x_v)$ for $w\in D_v$. Since there are exactly $\sum_{v\in K}|D_v|$ (linearly independent) equality constraints in L'_{CS}, the dimension of L'_{CS} is the same as that of L_{CS}. L'_{CS} includes more decision variables but solutions are projected onto the set of solutions that are feasible with respect to L_{CS} by the associated equality constraints. Thus, we conclude that $L_{CS} \equiv L'_{CS}$. **Q.E.D.**

From (27), we have $R_O = L_{CS} \bigcap\limits_{p\in P} C_{SP_p} = L_{CS} \bigcap\limits_{p\in P} conv(S_{SP_p})$. Let S'_{SP_p} denote the set of integral points that is equivalent to the corresponding to set of integral points S_{SP_p} in $|\bar{V}|$-dimensional space (i.e., $S'_{SP_p} \equiv S_{SP_p}$). Therefore, using $L_{CS} \equiv L'_{CS}$, R_O may be written as :

$$R_O \equiv L'_{CS} \bigcap_{p\in P} conv(S'_{SP_p}).$$

Since $L'_{CS} = L^{=}_{CS} \cap L^{\le}_{CS}$, $$R_O \equiv L^{=}_{CS} \cap L^{\le}_{CS} \bigcap_{p\in P} conv(S'_{SP_p}). \tag{29}$$

Relative to the edge-disjoint partitions formed in Cloning, edge inequalities in $\bar{E}_p$ comprise the block-diagonal set $p\in P$. Let $\bar{S}_{SP_p}$ denote the set of integral solutions that are feasible relative to block-diagonal set $\bar{E}_p$ for $p\in P$:

$$\bar{S}_{SP_p} = \left\{\mathbf{x}\in Z_+^{|\bar{V}|} : x_u + x_v \le 1 : (u,v)\in\bar{E}_p,\ x_v\in\{0,1\}\ \forall v\in\bar{V}\right\} \text{ and}$$

let $\overline{C}_{SP_p} = conv(\overline{S}_{SP_p})$. The block diagonal set $p \in P$ in Cloning ($\overline{E}_p$) incorporates the inequalities associated with edges in E_p as well as those associated with clones. $L_{CS}^{\leq}$ denotes the polytope corresponding to the inequalities associated with clones. A block diagonal set $p \in P$ incorporates a set of inequalities corresponding to clones that are added into p. In other words, block diagonal set p incorporates a subset of inequalities from the set (of inequalities) that defines $L_{CS}^{\leq}$ (i.e., $x_u + x_w \leq 1$ where $w \in D_v$ and $(u,v) \in \hat{E}$) for which $p = \pi_u = \pi_w$. Let $L_{CS_p}^{\leq}$ denote the polytope associated with inequalities (corresponding to clones) that are added to partition $p \in P$ such that $L_{CS}^{\leq} = \bigcap_{p \in P} L_{CS_p}^{\leq}$. $\overline{S}_{SP_p}$ consists of integer points, which are feasible with respect to edge inequalities E_p as well as inequalities corresponding to $\overline{L}_{CS_p}^{\leq}$. Let $S_{CS_p}^{\leq}$ denote the set of integer solutions that are feasible relative to $L_{CS_p}^{\leq}$. The feasible integer solutions with respect to a block-diagonal set p in Cloning may be written as:

$$\overline{S}_{SP_p} = S'_{SP_p} \bigcap S_{CS_p}^{\leq} . \tag{30}$$

$L_{CS}^{=}$ gives the polytope associated with the coordinating set in Cloning as it consists of equalities, each of which relates a cloned vertex with one of its clones. Let $\overline{L}_{CS}$ denote this polytope:

$$\overline{L}_{CS} = L_{CS}^{=} = \left\{ \mathbf{x} \in R_{+}^{|\overline{V}|} : x_w = x_v \ \forall \ v \in K, \ w \in D_v, \ 0 \leq x_v \leq 1 \ \ \forall v \in \overline{V} \right\}.$$

Let R_C denote the polytope formed by the intersection of $\overline{L}_{CS}$ and $\overline{C}_{SP_p}$:

$$R_C = \bar{L}_{CS} \bigcap_{p \in P} \bar{C}_{SP_p} \,. \tag{31}$$

Proposition 2: $R_C \subseteq R_O$.

Proof: From (30), we have $\bar{S}_{SP_p} = S'_{SP_p} \bigcap S^{\le}_{CS_p}$,

$\Rightarrow \quad conv(\bar{S}_{SP_p}) = conv(S'_{SP_p} \bigcap S^{\le}_{CS_p})$,

$\Rightarrow \quad conv(\bar{S}_{SP_p}) \subseteq conv(S'_{SP_p}) \bigcap conv(S^{\le}_{CS_p})$,

$\Rightarrow \quad conv(\bar{S}_{SP_p}) \subseteq conv(S'_{SP_p}) \bigcap L^{\le}_{CS_p}$, (as $conv(S^{\le}_{CS_p}) \subseteq L^{\le}_{CS_p}$),

$\Rightarrow \quad \bigcap_{p \in P} conv(\bar{S}_{SP_p}) \subseteq \bigcap_{p \in P} \left(conv(S'_{SP_p}) \bigcap L^{\le}_{CS_p} \right)$,

$\Rightarrow \quad \bigcap_{p \in P} conv(\bar{S}_{SP_p}) \subseteq \left(\bigcap_{p \in P} conv(S'_{SP_p}) \right) \bigcap \left(\bigcap_{p \in P} L^{\le}_{CS_p} \right)$,

$\Rightarrow \quad \bigcap_{p \in P} conv(\bar{S}_{SP_p}) \subseteq \left(\bigcap_{p \in P} conv(S'_{SP_p}) \right) \bigcap L^{\le}_{CS}$, (as $L^{\le}_{CS} = \bigcap_{p \in P} L^{\le}_{CS_p}$).

From (31), $R_C = \bar{L}_{CS} \bigcap_{p \in P} \bar{C}_{SP_p} = \bar{L}_{CS} \bigcap_{p \in P} conv(\bar{S}_{SP_p})$; and substituting for $\bigcap_{p \in P} conv(\bar{S}_{SP_p})$,

$$R_C = \bar{L}_{CS} \bigcap_{p \in P} conv(\bar{S}_{SP_p}) \subseteq \bar{L}_{CS} \bigcap L^{\le}_{CS} \bigcap_{p \in P} conv(S'_{SP_p}) \,,$$

$\Rightarrow \quad R_C \subseteq L^{=}_{CS} \bigcap L^{\le}_{CS} \bigcap_{p \in P} conv(S'_{SP_p})$, (Since $\bar{L}_{CS} = L^{=}_{CS}$).

Using (29), $R_C \subseteq R_O$. **Q.E.D.**

Finally, using $S \subseteq C \subseteq R_O \subseteq L$ from (28) and $R_C \subseteq R_O$, we have $S \subseteq C \subseteq R_C \subseteq R_O \subseteq L$. Let Z^*_L and Z^*_C denote the *optimal* solution values obtained by solving the MWISP objective function (1) on polytopes L and C, respectively.

Similarly, let $Z^*_{R_C}$ and $Z^*_{R_O}$ denote the optimal solution obtained by solving the MWISP on polytopes R_C and R_O , respectively. Thus,

$$Z^*_{MWISP} = Z^*_C \leq Z^*_{R_C} \leq Z^*_{R_O} \leq Z^*_L .$$

Proposition 3: In B&P search tree, Cloning gives tighter bound at the root node than the bound obtained by the OBP.

Proof: From (27) and (31), we have

$$R_C = \overline{L}_{CS} \bigcap_{p \in P} \overline{C}_{SP_p} \text{ and } R_O = L_{CS} \bigcap_{p \in P} C_{SP_p} .$$

If we apply DWD to the constraint set of R_O, the constraints that form L_{CS} are relegated to form the constraints in the RMP of the OBP model (see (2)-(5)) and those that form C_{SP_p} create the constraint set for the sub-problem (see (6)). Similarly if we apply DWD to the constraint set of R_C, the constraints in $\overline{L}_{CS}$ form the constraints in the RMP of the Cloning model and those in $\overline{C}_{SP_p}$ creates the constraint set for sub-problem. Since $Z^*_{R_C} \leq Z^*_{R_O}$, it implies that Cloning gives tighter bound at the root node than the bound obtained by the OBP model. **Q.E.D.**

However, should sub-problems exhibit the Integrality Property, (i.e., all extreme points of L_{SP_p} for $p \in P$ are integral),

$$Z^*_{R_C} = Z^*_{R_O} = Z^*_L .$$

Hence, to obtain a tighter bound, it is imperative that sub-problems avoid the Integrality Property.

CHAPTER V

IMPLEMENTATION ISSUES

Cloning involves two key issues: (a) Selecting vertices to be cloned, (b) Assigning weights to clones. We discuss these issues and propose solutions in this chapter. We present the overall algorithmic steps involved in solving the MWISP by our B&P approach and introduce a new concept, *Partial Cloning*, developed to exploit the desirable virtues of both OBP and Cloning approaches.

5.1 Selecting Vertices for Cloning

Each vertex that is cloned increases the size of the partition (i.e., sub-problem) into which it is cloned as well as the number of equality constraints (in the RMP). Especially in dense graphs, Cloning may add a large number of vertices, resulting in larger sub-problems that are more difficult to solve. Thus, it is imperative that Cloning duplicate the minimum number of vertices. For example, in Figure 2, to replace edges (v_2, v_3) and (v_6, v_3), either v_2 and v_6 could be cloned into partition 2, increasing its size by two vertices (and two edges) or v_3 could be cloned into partition 1, increasing its size by only one vertex (and, of course, two edges). This issue can be resolved by solving an appropriate set covering problem. Using binary decision variables $y_{vp} = 1$ if vertex v is cloned into partition p and $y_{vp} = 0$ otherwise, the set covering problem may be formulated as follows:

$$Z = Min\left\{\sum_{v\in\hat{V}}\sum_{p\in P} y_{vp} : \mathbf{y} \in Q_{sc} \cap B^m\right\}, \qquad (32)$$

where

$$Q_{sc} = \left\{ \mathbf{y} \in R_+^m : y_{uq} + y_{vp} \geq 1 \ \forall uv \in \hat{E},\ u \in V_p \text{ and } v \in V_q \right\} \text{ and } m = |\hat{V}|\,(|P|-1).$$

The set covering problem is NP-Hard [12], but a "near optimal" solution would suit our purpose so we propose a modified version of the greedy set covering heuristic [19] to quickly obtain a solution. We refer to this heuristic as the modified set covering heuristic:

Step 1: $i = 0$, $K = \varnothing$, $D_v = \varnothing \ \forall\ v \in \hat{V}$; $\overline{V}_p = V_p$ and $\overline{E}_p = E_p$ for all $p \in P$.

Step 2: For every vertex $v \in \hat{V}$, determine $N_p(v)$ from $\hat{E}$.

Step 3: Calculate $N_{p^*}(v^*)$ = max $\left\{ \left|N_p(v)\right| : p \in P, v \in \hat{V}, \pi_v \neq p \right\}$. $N_{p^*}(v^*)$ identifies the vertex v^* to be cloned as the one adjacent to the largest number of vertices not in the same partition (i.e., π_v) and the partition p^* into which it would be cloned. Clone vertex v^* into partition p^* as vertex w.

Step 4: Update; $D_v \leftarrow D_v \bigcup \{w\}$, $\overline{V}_p \leftarrow \overline{V}_p \cup \{w\}$, $\overline{E}_p \leftarrow \overline{E}_p \cup \{(u,w) : u \in N_{p^*}(v^*)\}, K \leftarrow K \cup \{v\}$, $\hat{E} \leftarrow \hat{E} \setminus \{(u,v^*) : u \in N_{p^*}(v^*)\}$ and $\hat{V} = \hat{V} \setminus v$ if $N_p(v) = \varnothing \ \forall\ p \in P$, $\pi_v \neq p$.

Step 5: Repeat Steps 2, 3 and 4 until $\hat{V} = \varnothing$.

Step 6: For $v \in K$, $w \in D_v$ are the clones of vertex v.

5.2 Assigning Weights

Appropriate weights must be assigned to a cloned vertex and its clones. To be an exact copy, a clone should have the same weight as that of its originating vertex but this would increase the total weight in the graph so that the optimal solution to the MWISP on the graph with clones would not be the same as that on the original graph. We implemented two strategies that result in total weights that are the same in both the original graph and the one that results from cloning. One strategy is to divide the weight of an originating vertex equally among the set of copies. Another, and in fact the simplest, strategy is to assign a null-weight to clones. The chapter on computational evaluation compares the impacts of these strategies on run-time.

5.3 Solving the MWISP

Cloning may be detailed as follows:

Step 1: Partition an original graph into $|P|$ partitions using METIS [15, 16, 17].

Step 2: Apply the modified set covering heuristic to select the set of vertices to be cloned and identify the clones for each. Update the RMP to include equalities corresponding to equivalence relationships between each originating vertex and its clones. Update sub-problems to include clones ($w \in D_v$) and their associated edge inequalities.

Step 3: Solve the Cloning formulation utilizing the MCP algorithm to solve sub-problems. At each iteration, re-optimize RMP over "known" columns and use the resulting dual variables to define the objective function coefficients of decision variables in sub-problems. Use a pool to store the columns generated by the sub-problems. Maintain previously generated columns in the pool and optimize over these “known” columns before solving sub-problems in an attempt to conserve run-time. Branch on clique inequalities as described in Warrier et al [25].

5.4 Partial Cloning

Warrier et al [25] observed that the OBP results in large $|\hat{E}|$ so that the RMP may comprise a large number of constraints and require a lengthy solution time. Cloning decreases the number of RMP constraints because the modified set covering heuristic (Section 5.1) seeks the minimum number of vertices to clone. On the other hand, this approach adds clones to partitions, increasing the size of individual sub-problems and making them more challenging for the MCP algorithm to solve. Hence, Cloning introduces a trade off by which problem complexity can be distributed among the RMP and sub-problems.

The sizes of the partitions (sub-problems) can be controlled to some extent by specifying the number of partitions that METIS is required to develop. However, the $\overline{V}_p$ and $\overline{E}_p$ depend on the characteristics of partitions created by METIS and the set of clones prescribed by the modified set covering heuristic.

We propose a new approach to achieve a favorable trade-off between the size (and tightness) of the RMP and the sizes of the sub-problems. This approach, which we call *Partial Cloning*, may not clone all vertices in $\hat{E}$, perhaps retaining some edge inequalities in the RMP. We update step (2) of the modified set covering heuristic (Section 5.3) to implement Partial Cloning by setting a threshold (*PCThreshold*) to affect the vertex selected for cloning. To implement Partial Cloning, Step 2 in the heuristic given in chapter 5.1 is updated to be:

Step 2 (updated): If $\max\left\{ \left| N_p(v) \right| : p \in P, v \in \hat{V} \right\} > PCThreshold$, continue to Step 3, else go to Step 5.

This modification allows the RMP to retain some edge inequalities while including equalities associated with clones. Henceforth, we use Complete Cloning (CC) to specify the approach where all the edges in $\hat{E}$ are relocated by cloning and use Partial Cloning (PC) to specify the approach in which only a subset of edges in $\hat{E}$ are relocated. We set *PCThreshold* to 1 in our tests so the modified set covering heuristic adds clones corresponding to those vertices $v \in \hat{V}$ and partition $p \in P$, for which $|N_p(v)| > 1$ for $\pi_v \neq p$. If *PCThreshold* is set to 0, Complete Cloning results, yielding larger, more sparse sub-problem that are more challenging for MCP algorithm to solve.

CHAPTER VI

COMPUTATIONAL EVALUATION

We compare CC, PC, OBP and MCP computationally using two types of instances: (1) DIMACS Instances taken from the Second DIMACS Implementation Challenge [14], and (2) random p- graphs: These random graphs are generated by specifying the number of vertices $|V|$ and value p (probability that edge (u,v) is included in the graph). We conducted all tests on a Dell PC with a 3.06 GHz Pentium IV processor and 512 MB of memory using the Visual C++ environment and CPLEX 7.1.

Preliminary testing of the two Cloning approaches (CC and PC) each using the two weight-assignment strategies (Chapter 5.2) showed that assigning null weights to clones performs better than assigning each clone the same weight associated with its originating vertex. Hence, we presents results that assign null weights to all clones.

We select $|P|$ based on the criterion that the resulting sub-problem, after partitioning and cloning, should be less challenging for MCP to solve. However, there is no definite way to ascertain the size of sub-problems that will result from Cloning. Preliminary tests showed that, for graphs with 100 or more vertices and edge densities less than 40%, $|P| \geq 6$ results in sub-problems that MCP can solve effectively and for graphs having edge densities greater than 40%, $|P| = 2$ or 3 results in sub-problems that MCP can solve effectively.

The Partial Cloning parameter, *PCThreshold*, affects the mix of equalities and inequalities in the RMP. We set the default value of *PCThreshold* to 1. On some instances, a value of 1 leads to as many clones as in CC (because for all $v \in \hat{V}$, $|N_p(v)|>1 \quad p \in P$, $\pi_v \neq p$). Hence, in these cases, *PCThreshold* is set to $|\hat{E}|/$ M, where M is equal to $\sum_{v \in K} |D_v|$. $|\hat{E}|/$ M gives the average number of vertices that a vertex $v \in \hat{V}$ is connected by edges $(u,v) \in \hat{E}$.

Table 1 compares the performances of the three B&P approaches (OBP, CC and PC) and MCP in application to the DIMACS instances. Performance measures include the number of constraints in RMP, optimal solution at root node of the B&B tree, and computational time(in cpu seconds). The first five columns give the name of instance; number of vertices, $|V|$; density; Z^*_{MWISP}; and number of partitions, $|P|$. Columns 6-8 give the number of equality constraints in the RMP for OBP, CC, and PC, respectively (the number in the braces give the number of inequalities in the RMP corresponding to edge inequalities). In OBP, RMP comprises only inequalities, and in CC, RMP comprises only equalities. In PC, RMP comprises a mix of inequality and equality constraints. Columns 9-11 give Z^*_{LP} (OBP), Z^*_{LP} (CC) and Z^*_{LP} (PC), the optimal solution at the root node (of B&B tree) for OBP, CC and PC, respectively. The optimal solution at root node gives an upper bound on Z^*_{MWISP}. Computational results confirm that CC and PC give upper bounds that are tighter than the one that OBP gives and, as expected, Z^*_{LP} (CC) $\leq$ Z^*_{LP} (PC) $\leq$ Z^*_{LP} (OBP).

Table 1 Performance measures for DIMACS instances.

Graph	|V	Density	Z_{IP}	|P|	Number of RMP			Z_{LP}			Time (in sec)			
					OBP	C1	C2	OBP	C1	C2	OBP	C1	C2	WCP
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
johnson824comp	28	44	4	2	(64)	13	1(56)	5.75	4	5	0.38	0.2	0.27	
johnson824comp	28	44	4	3	(103)	29	25(4)	8	4.25	4.25	0.75	0.13	0.063	0.02
johnson824comp	28	44	4	4	(100)	39	29(10)	6.5	4	4.5	0.52	0.09	0.094	
johnson824comp	28	44	4	5	(124)	45	39(6)	9.5	4.5	4.5	1.38	0.11	0.078	
johnson844comp	70	23	14	3	(240)	69	59(10)	15	14	14	1.02	78.33	22.87	
johnson844comp	70	23	14	5	(374)	127	93(34)	21.25	14	14	19.47	52.97	6.66	14.89
johnson844comp	70	23	14	8	(381)	161	104(57)	17.5	14	14	10.34	18.22	1.39	
manna9comp	45	7	16	2	(10)	9	1(8)	17.67	17.6	17.6	9.63	29.83	8.422	
manna9comp	45	7	16	3	(16)	14	2(12)	18	17.75	17.75	1.38	4.66	1.187	
manna9comp	45	7	16	4	(25)	19	6913)	20	17.86	18.5	3.14	3.67	0.781	620.97
manna9comp	45	7	16	5	(26)	21	5916)	19	18	18	1.31	1.53	0.469	
manna9comp	45	7	16	6	(29)	23	6(17)	19.5	18	18	2.41	1.36	0.469	
cfat2001comp	200	92	12	2	(8999)	97	29(6253)	13.5	12	13	2.13	0.72	2.03	0.11
cfat2001comp	200	92	12	3	(12137)	196	87(6575)	15	12	12.33	10.44	3.1	8.08	
cfat2002comp	200	84	24	2	(7952)	97	30(5404)	30	24	27.5	14.3	544.69	16.13	0.42
hamming62comp	64	10	32	4	(64)	64	0(64)	32	32	32	0.22	135.77	0.22	
hamming62comp	64	10	32	5	(101)	67	27(40)	32	32	32	0.19	17.05	0.89	71.75
hamming62comp	64	10	32	6	(114)	75	33(42)	32	32	32	0.42	21.66	1.25	
hamming82comp	256	3	128	20	(626)	-	95(414)	128	-	128	3.3	-	27.05	-
johnson1624comp	120	23	8	10	-	-	253(162)	-	-	8.5	-	-	23.74	-

In fact, Z^*_{LP}(CC) $= Z^*_{MWISP}$ for most of the instances (giving an integrality gap of 0%). Furthermore, with an increase in $|P|$, the bound gets weaken (integrality gap increases) for each of the three B&P approaches. Figures 5 and 6 shows variation of Z^*_{LP}(OBP), Z^*_{LP}(CC) and Z^*_{LP}(PC) with increase in $|P|$ for "manna9comp" and "johnson824comp" respectively.

Columns 12-14 compare the run times (cpu seconds) required by OBP, CC and PC to solve each instance, excluding the times required for partitioning and cloning, which are trivial. Column 15 gives the run time required to solve each instance by MCP. A "-" indicates that the corresponding instance requires more than 12 hours of run time. We found that, as $|P|$ increases, the run time required by each B&P approach to solve an instance varies depending upon whether the instance is dense or sparse. For dense instances, run time increases with an increase in $|P|$ and, for sparse instances run time first decreases and then increases as $|P|$ increases, so some value of $|P|$ gives minimum run-time for sparse graphs. We vary the value of $|P|$ for a few representative instances (e.g., manna9comp, johnson824 comp) to show the variation in run-time as $|P|$ increases. For the remaining instances, we tabulate results for those $|P|$ that give minimum run-time (for e.g., we set $|P|$ = 10 for johnson1624comp and $|P|$ = 20 for hamming82comp). PC gives quite competitive results for most of the DIMACS instances. Figures 7 and 8 shows variation of run time for three B&P approaches with increase in $|P|$ for "manna9comp" and "johnson824comp" respectively.

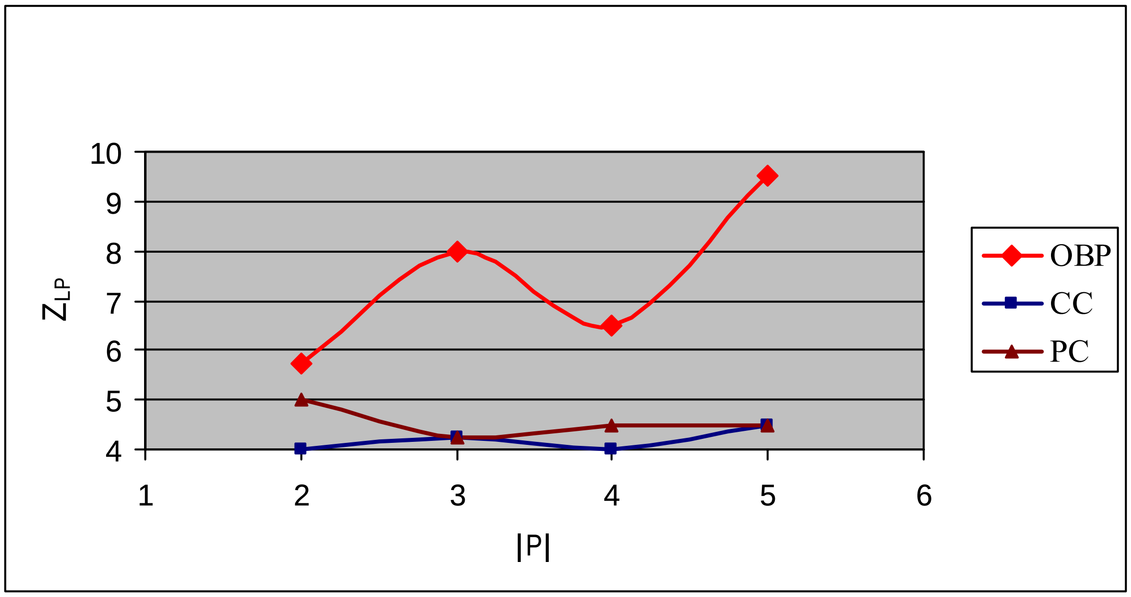

Fig. 5 Z_{LP} vs |P| for johnson824comp.

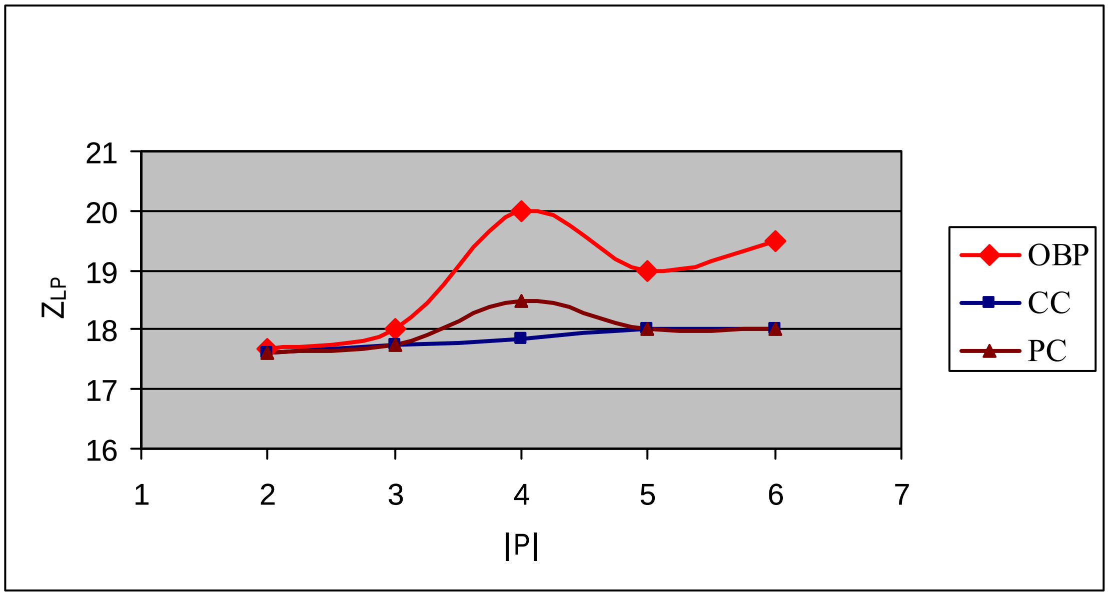

Fig. 6 Z_{LP} vs |P| for manna9comp.

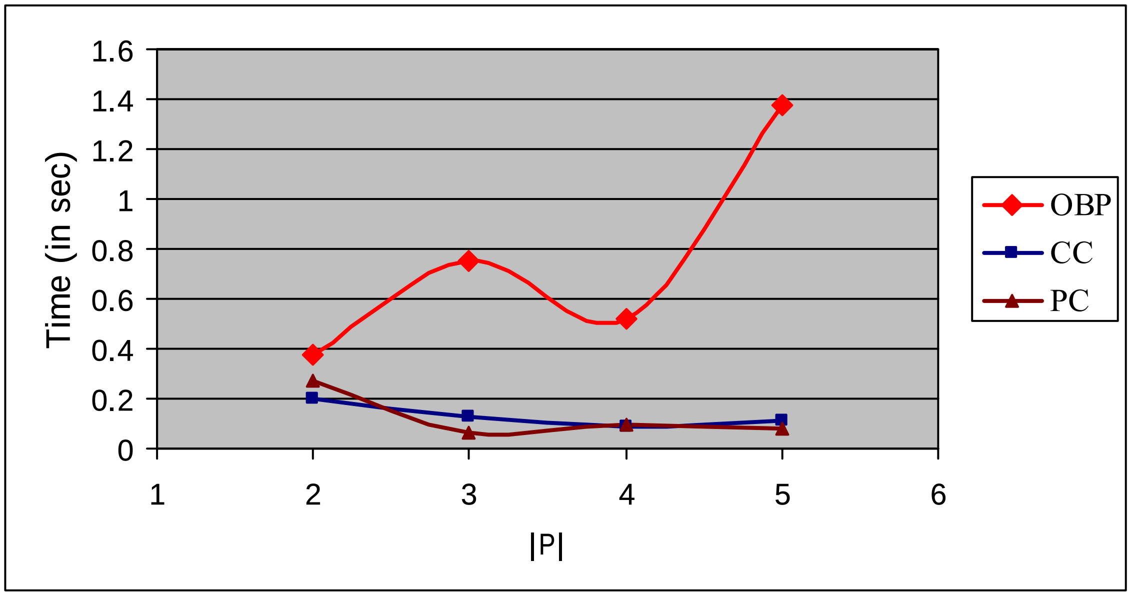

Fig. 7 Run time vs |P| for Johnson 824 comp.

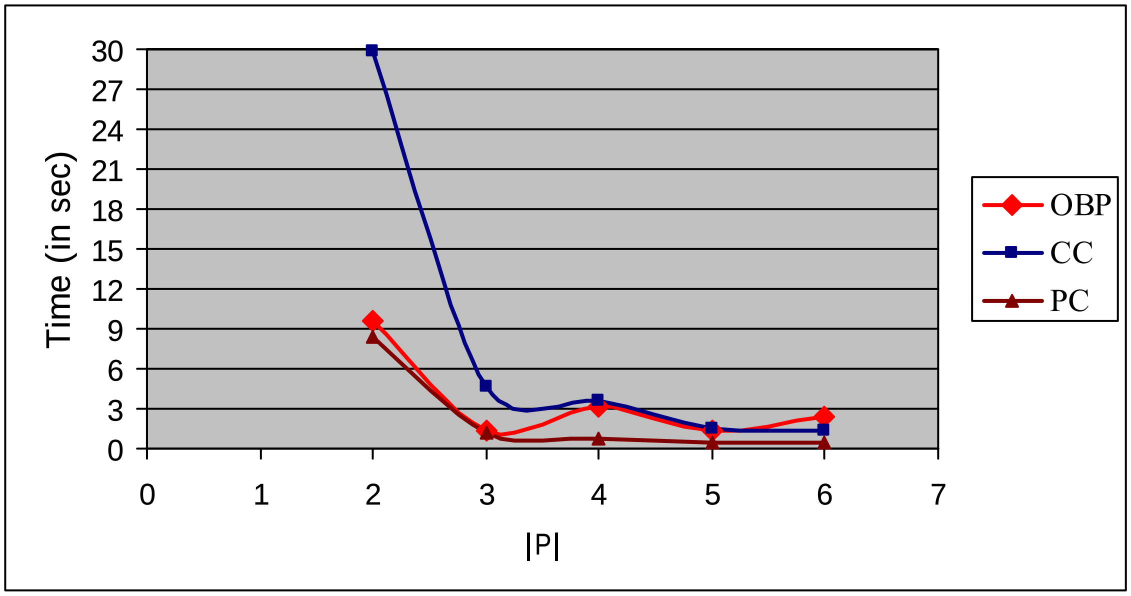

Fig. 8 Run time vs |P| for manna9comp.

Table 2 reports application of the three B&P approaches to random p-graphs, using the same column headings. “W0” in the name of instance indicates an un-weighted graph and “W1” indicates a weighted graph. Run times reported in columns 12-14 of Table 2 show that MCP outperforms all three B&P approaches on random instances having densities greater than 40%. For instances with densities below 20%, all three B&P approaches perform better than MCP. Comparing run times in columns 12-14 shows that weighted graphs are generally less challenging to solve than un-weighted graphs. Although CC never gives the best run-time, it gives quite competitive results for highly dense and highly sparse instances. Furthermore, as observed in DIMACS instances, for all the random graphs, we have Z_{LP}^*(CC) ≤ Z_{LP}^*(PC) ≤ Z_{LP}^*(OBP).

To gain further insight into the performance of B&P approaches for solving the MWISP, we compare several additional performance measures in Tables 3 and 4, which relates to the instances reported in Tables 1 and 2. Columns 1 and 2 in Tables 3 and 4 give the name of the instance and the number of partitions, $|P|$, respectively. Columns 3-5 give number of RMP iterations required and columns 6-8 give number of nodes explored in the B&B tree to obtain an optimal integral solution by each of the three B&P approaches. If the number of nodes explored is zero, the optimal integer solution was obtained at root node of the B&B search tree (i.e., Z_{LP}^*(CC) = Z_{MWISP}^*). PC typically explores a number of B&B nodes that is between the numbers of nodes required by CC and OBP.

Table 2 Performance measures for p-graphs.

Graph	|V|	Density	Z_{IP}	|P|	number of RMP			Z_{LP}			Run time (in sec)			
		(%)			OBP	CC	PC	OBP	CC	PC	OBP	CC	PC	WCP
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
RG_GV1_W1_P05	100	5	481	10	(150)	111	32(79)	529.50	496.17	501.70	20.73	70.33	5.67	-
RG_GV1_W0_P10	100	10	30	10	(349)	222	97(125)	41.25	34.57	36.52	283.47	4344.48	463.27	-
RG_GV1_W1_P10	100	10	371	10	(349)	222	97(125)	478.00	402.25	422.87	68.77	2819.42	41.25	-
RG_GV1_W0_P20	100	20	19	10	(789)	-	234(122)	-	-	24.88	-	-	571.82	566.61
RG_GV1_W1_P20	100	20	246	10	(789)	-	234(122)	399.00	-	289.63	273.15	-	2549.91	
RG_GV1_W0_P30	100	30	15	10	(1231)	-	341(105)	31.50	-	18.79	677.49	-	1434.20	36.25
RG_GV1_W1_P30	100	30	193	10	(1231)	-	341(105)	-	-	219.38	-	-	583.49	
RG_GV1_W0_P40	100	40	12	6	(1517)	-	254(14)	21.50	-	12.00	139.43	-	891.23	3.89
RG_GV1_W1_P40	100	40	161	6	(1517)	268	254(14)	272.50	161.00	161.00	86.90	935.03	759.63	
RG_GV1_W0_P50	100	50	9	2	(1089)	49	32(320)	12.75	9.00	11.00	19.30	141.14	51.55	0.88
RG_GV1_W1_P50	100	50	120	2	(1089)	49	32(320)	170.00	120.00	141.00	15.34	112.92	34.73	
RG_GV1_W0_P60	100	60	7	2	(1325)	49	32(399)	10.00	7.00	9.50	13.55	11.79	14.41	0.30
RG_GV1_W1_P60	100	60	52	2	(1325)	49	32(399)	68.20	52.00	61.00	6.02	29.87	8.55	
RG_GV1_W0_P60	100	60	7	4	(2067)	150	132(182)	16.00	7.00	8.22	83.28	27.04	40.75	
RG_GV1_W1_P60	100	60	52	4	(2067)	150	132(182)	98.50	52.00	52.00	40.67	103.69	27.21	
RG_GV1_W0_P70	100	70	7	2	(1573)	49	37(328)	9.00	7.00	7.67	1.89	7.34	3.27	0.94
RG_GV1_W1_P70	100	70	41	2	(1573)	49	37(328)	60.00	41.00	48.33	5.27	6.56	4.02	
RG_GV1_W0_P70	100	70	7	4	(2444)	151	139(148)	12.00	7.08	7.43	24.03	25.93	14.95	
RG_GV1_W1_P70	100	70	41	4	(2444)	151	139(148)	77.00	43.41	46.07	31.21	19.14	13.14	
RG_GV1_W0_P80	100	80	5	2	(1875)	50	36(478)	7.67	5.00	6.50	3.13	1.25	1.22	0.03
RG_GV1_W1_P80	100	80	38	2	(1875)	50	36(478)	51.50	38.00	41.67	1.41	1.15	1.11	
RG_GV1_W0_P80	100	80	5	4	(2878)	152	140(156)	9.50	5.00	5.67	16.47	4.92	5.70	
RG_GV1_W1_P80	100	80	38	4	(2878)	152	140(156)	68.00	38.00	38.00	12.03	5.13	2.25	
RG_GV1_W0_P90	100	90	4	2	(2159)	49	43(235)	6.00	4.00	4.85	1.48	1.39	0.44	0.02
RG_GV1_W1_P90	100	90	32	2	(2159)	49	43(235)	43.00	32.00	34.60	1.02	0.33	0.44	
RG_GV1_W0_P90	100	90	4	4	(3272)	152	118(622)	8.80	4.25	5.55	8.31	2.85	2.69	
RG_GV1_W1_P90	100	90	32	4	(3272)	152	118(622)	63.50	32.00	41.75	7.09	0.73	1.72	

Table 3 Additional performance measures for DIMACS instances.

Graph	|P|	number of RMP iterations			number of B&B nodes			% of time to clone		% of time to obtain Z_{LP}			% of time to solve sub-problems		
		OBP	CC	PC	OBP	CC	PC	CC	PC	OBP	CC	PC	OBP	CC	PC
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
johnson824comp	2	320	23	319	34	0	35	4.01	0.00	33.33	91.63	43.08	37.33	41.87	45.51
johnson824comp	3	981	409	66	98	0	0	0.00	4.65	2.13	87.20	100.00	54.00	49.60	76.19
johnson824comp	4	746	65	67	56	0	0	1.62	9.01	2.91	84.04	82.97	30.29	67.02	65.96
johnson824comp	5	1906	116	97	169	0	0	4.76	5.83	2.25	86.24	79.48	29.67	72.48	0.00
johnson844comp	3	260	407	185	14	0	0	0.02	0.27	13.78	99.98	100.00	73.82	98.52	98.69
johnson844comp	5	11172	664	265	399	0	0	0.14	1.56	0.96	99.98	99.78	28.93	85.86	83.63
johnson844comp	8	7648	645	230	231	0	0	0.73	6.44	0.61	99.71	97.77	15.56	58.87	36.30
manna9comp	2	291	623	283	24	21	21	0.00	0.00	34.10	26.87	37.11	98.88	99.53	99.07
manna9comp	3	480	1258	512	24	32	27	0.00	1.11	20.44	18.45	26.28	87.49	89.54	83.99
manna9comp	4	3062	1889	591	143	37	22	0.49	0.00	4.97	16.61	29.96	65.55	74.13	70.04
manna9comp	5	2260	1603	754	99	30	24	0.93	0.00	5.94	9.21	13.43	52.40	52.91	40.51
manna9comp	6	4368	1495	809	168	28	24	1.00	0.00	2.62	11.55	9.81	45.93	59.16	49.68
cfat2001comp	2	61	24	60	5	0	5	38.82	15.80	18.40	97.93	23.80	5.79	97.93	8.50
cfat2001comp	3	244	114	167	18	0	0	35.83	17.00	6.43	98.45	39.53	1.63	92.39	7.10
cfat2002comp	2	351	101	948	20	0	19	0.50	2.20	2.62	99.99	27.51	13.99	99.98	43.08
hamming62comp	4	40	256	40	0	0	0	0.02	0.00	93.98	99.99	100.00	87.97	99.61	85.39
hamming62comp	5	68	585	90	0	0	0	0.18	1.53	91.49	100.00	98.31	41.49	90.31	87.98
hamming62comp	6	111	524	91	0	0	0	0.22	0.87	96.21	99.93	98.08	55.92	92.79	92.06
hamming82comp	20	329	-	860.00	0	-	0.00	-	2.40	99.92	-	93.93	16.45	-	14.15
johnson1624comp	10	-	-	355	-	-	0.00	-	2.89	-	-	99.82	-	-	89.68

Table 4 Additional performance measures for random p-graphs.

Graph	|P|	number of RMP iterations			number of B&B nodes			% of time to clone		% of time to obtain Z_{LP}			% of time to solve sub-problems		
		OBP	CC	PC	OBP	CC	PC	CC	PC	OBP	CC	PC	OBP	CC	PC
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
RG_GV1_W1_P05	10	6516	7339	2646	81	15	25	0.18	1.04	1.73	12.90	9.38	66.01	49.61	50.39
RG_GV1_W0_P10	10	157482	103880	129175	2474	132	890	0.01	0.10	0.13	3.53	0.71	11.84	42.63	21.85
RG_GV1_W1_P10	10	30271	76771	10661	403	78	58	0.01	0.45	0.59	3.74	5.61	27.59	35.74	30.36
RG_GV1_W0_P20	10	-	-	242987	-	-	857	-	0.02	-	-	1.52	-	-	27.11
RG_GV1_W1_P20	10	91948	-	49898	1897	-	145	-	0.08	0.15	-	5.03	8.53	-	31.35
RG_GV1_W0_P30	10	145002	-	48455	4009	-	83	-	0.05	0.07	-	9.21	6.76	-	39.58
RG_GV1_W1_P30	10	-	-	15702	-	-	23	-	0.23	-	-	21.17	-	-	45.36
RG_GV1_W0_P40	6	31227	-	1059	1210	-	0	-	0.06	0.17	-	99.99	7.64	-	97.54
RG_GV1_W1_P40	6	18139	1388	1107	633	0	0	0.06	0.07	0.31	99.99	99.99	7.40	95.44	98.28
RG_GV1_W0_P50	2	2826	528	1355	213	3	34	0.09	0.44	5.10	67.22	41.65	53.99	99.51	90.54
RG_GV1_W1_P50	2	3026	231	1721	221	0	39	0.11	0.24	2.85	99.98	30.62	40.12	99.66	93.05
RG_GV1_W0_P60	2	2405	110	1694	212	0	60	1.12	1.01	1.96	99.87	29.17	34.70	98.80	78.53
RG_GV1_W1_P60	2	947	222	700	77	0	19	0.51	1.53	7.53	99.65	52.11	46.55	99.11	85.04
RG_GV1_W0_P60	4	14120	401	1321	958	0	5	1.67	1.33	0.21	99.88	58.02	10.83	91.61	84.20
RG_GV1_W1_P60	4	6434	1029	433	382	0	0	0.43	1.76	0.50	99.98	99.94	9.20	93.66	90.35
RG_GV1_W0_P70	2	318	163	228	27	0	4	2.13	4.61	6.61	99.79	61.24	30.05	97.67	82.79
RG_GV1_W1_P70	2	1012	150	580	97	0	20	2.58	4.25	2.07	99.75	47.06	21.98	97.85	76.66
RG_GV1_W0_P70	4	3673	3064	1385	247	8	7	2.01	4.23	0.84	35.85	57.26	4.89	61.32	58.60
RG_GV1_W1_P70	4	4871	2453	2051	344	7	16	2.71	4.64	0.60	37.87	32.87	4.89	59.22	50.48
RG_GV1_W0_P80	2	573	94	303	60	0	16	10.55	9.72	2.50	98.72	28.21	13.95	93.76	42.32
RG_GV1_W1_P80	2	244	90	188	25	0	6	10.95	12.03	6.69	98.62	59.09	21.19	90.68	57.47
RG_GV1_W0_P80	4	2266	360	736	185	0	5	11.12	10.94	1.23	79.16	54.51	4.35	58.51	44.64
RG_GV1_W1_P80	4	1638	463	273	128	0	0	10.32	21.03	1.43	99.68	97.91	3.91	80.34	65.86
RG_GV1_W0_P90	2	178	81	59	12	0	0	10.64	18.57	4.18	96.62	96.35	22.04	83.10	56.85
RG_GV1_W1_P90	2	174	64	145	20	0	6	23.18	23.32	6.10	95.12	42.69	9.15	61.58	60.73
RG_GV1_W0_P90	4	1060	315	543	102	0	9	24.08	18.34	3.38	98.35	52.90	3.59	31.62	14.99
RG_GV1_W1_P90	4	886	260	412	81	0	7	35.68	26.48	3.09	97.80	33.62	3.98	48.64	26.41

Columns 9-10 give the percentage of computational time spent in Cloning relative to the total run time. Columns 11-13 give the percentage of time utilized to obtain optimal solution at root-node relative to the total time spent in obtaining Z^*_{MWISP}. Results show that, for CC and PC, more than 90% of run time is spent in obtaining root-node solutions, Z^*_{LP} (CC) and Z^*_{LP} (PC). Columns 14-16 give the percentage of time required to solve sub-problems using the MCP algorithm relative to the time spent in prescribing an integral optimal solution.

Results show that OBP spends a smaller percentage of run time to solve sub-problems than CC and PC. OBP leads to sub-problems that are less challenging for MCP to solve, but gives a weak Z^*_{LP} (OBP) bound (see Tables 1 and 2). Because the upper bound is weak, OBP requires exploration of more nodes in the B&B search tree. (see Columns 6-8 in Tables 3 and 4) increasing the number of times the RMP is optimized, and, hence, the total number of RMP iterations (see Columns 3-5 in Tables 3 and 4). As a result, OBP spends most of the time optimizing the RMP and relatively little time solving sub-problems. In contrast, both CC and PC spend a considerable percentage of run time solving sub-problems. Cloning may increase the size of sub-problems dramatically, making them challenging for MCP but giving tighter bounds. Because upper bounds Z^*_{LP} (CC) and Z^*_{LP} (PC) are tight, CC and PC both explore fewer B&B nodes and, thus, require less run time to optimize.

Further, we compare the three B&P approaches in application to random p-graphs with densities less than 10%. Table 5 gives results for these graphs with the same column headings used in Tables 1 and 3. For CC, run time increases rapidly with an increase in graph density. However, the Z_{LP}^{*} (CC) bound is better than Z_{LP}^{*} (PC) and Z_{LP}^{*} (OBP) for all the test cases.

We conjecture that Cloning may work better in application to instances for which the resulting sub-problems are less challenging for MCP to solve. Increasing the value of $|P|$ may result in desirable sub-problems but weakens the upper bound and increases the overall run time by increasing the number of B&B nodes (which increases the time spent in optimizing the RMP).

Table 5 Performance measures for sparse random p-graphs.

Graph	|V|	Density	Z_{IP}	|P|	number of RMP constraints			Z_{LP}			run time (in sec)			
		(%)			OBP	CC	PC	OBP	CC	PC	OBP	CC	PC	WCP
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
RG_GV1_W1_P0.02	100	2	656	10	(30)	25	5(20)	656.00	656.00	656.00	0.25	1.03	0.36	-
RG_GV1_W1_P0.03	100	3	575	10	(70)	61	6(55)	575.00	575.00	575.00	0.17	1.63	0.19	-
RG_GV1_W1_P0.04	100	4	521	10	(108)	91	17(74)	534.00	525.20	526.67	3.50	24.19	1.64	-
RG_GV1_W1_P0.05	100	5	481	10	(150)	111	32(79)	529.50	496.17	501.71	17.92	67.95	5.16	-
RG_GV1_W1_P_0.06	100	6	448	10	(179)	136	39(97)	504.50	469.50	487.50	9.72	108.12	10.36	-
RG_GV1_W1_P_0.07	100	7	432	10	(248)	163	64(99)	521.50	457.00	469.50	26.05	583.67	26.84	-
RG_GV1_W1_P_0.08	100	8	406	10	(273)	180	70(110)	517.50	429.50	449.33	63.46	349.80	37.90	-
RG_GV1_W1_P_0.09	100	9	386	10	(297)	-	77(121)	479.00	-	431.65	61.35	-	36.23	-

CHAPTER VII

SUMMARY, CONCLUSIONS AND FUTURE RESEARCH

This thesis contributes a new vertex cloning approach to solve the MWISP within a B&P framework. This thesis achieves its objectives: formulation of the Vertex Cloning approach, analysis showing that Vertex Cloning yields a tighter formulation, effective methods to implement Vertex Cloning, and analysis of the computational efficacy of Vertex Cloning. This thesis also presents a variant of Cloning, Partial Cloning, which results in a mix of inequalities and equalities in the RMP. We compared the three B&P approaches on DIMACS instances as well as random p-graphs.

The B&P approach for solving the MWISP is built on the basic idea of decomposing the graph into smaller sub-graphs that are less challenging to solve. Warrier et al [25] developed their B&P approach for the MWISP by creating vertex-disjoint sub-graphs. Cloning enhances this idea by creating clones of vertices to convert a vertex-disjoint partition into an edge-disjoint partition. Cloning improves the OBP approach by creating a smaller RMP that gives tighter upper bounds at nodes of the B&B tree.

Cloning provides excellent bounds for the MWISP, but it may require lengthy runtime because it leads to larger sub-problems that may be more challenging to solve. In contrast, the RMP associated with OBP gives a weak bound but the approach requires less total run time because its sub-problems are smaller and can be solved more effectively. PC results in somewhat less challenging sub-problems than does CC and it gives tighter bounds than does the OBP. Consequently, PC solves MWISP effectively on both dense and sparse graphs.

Future research in this area could be directed towards developing efficient methods to create smaller edge-disjoint partitions that can be solved effectively. In addition, developing a more capable algorithm to solve sub-problems would help to reduce total run time.

REFERENCES

[1] V.E. AlekSeev, Polynomial algorithm for finding the largest independent sets in graphs without forks, Discrete Applied Mathematics 135 (2004) 3-16.

[2] E. Balas, C.S. Yu , Finding a maximum clique in an arbitrary graph, SIAM Journal on Computing 15 (4) (1986) 1054-1068.

[3] E. Balas, J. Xue , Minimum weighted coloring of triangulated graphs, with applications to maximum weight vertex packing and clique finding in arbitrary graphs, SIAM Journal on Computing 20 (2) (1991) 209-221.

[4] C. Barnhart, E. Johnson, G. Nemhauser, M. Savelsbergh, P. Vance, Branch-and-price: column generation for solving huge integer programs, Operations Research 46 (1998) 316-329.

[5] M.S. Bazaraa, J. J. Jarvis, H.D. Sherali, Linear Programming and Network Flows, 2nd edition, John Wiley and Sons, New York, 1990.

[6] I.M. Bomze, M. Budinich, P.M. Pardalos, M. Pelilo, The maximum clique problem, in: D.Z. Du, P.M. Pardalos (Eds.), Handbook of Combinatorial Optimization, Kluwer Academic Publishers, Boston, 1999, pp. 1-74.

[7] S. Butenko, P. Pardalos, I. Sergienko, V. Shyloand, P. Stetsuyk, Maximum independent sets in graphs arising from coding theory, in: Proceedings of the Seventeenth ACM Symposium on Applied Computing, Madrid, Spain, 2002, pp. 542-546.

[8] R. Carr, G. Lancia S. Istrail, Branch and cut algorithms for independent set problems: integrality gap and an application to protein structure alignment, Technical Report SAND2000-2171, Sandia National Labs, Livermore, CA, 2000.

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

[10] T.A. Feo, M.G.C. Resende, S.H. Smith, A greedy randomized adaptive search procedure for maximum independent set, Operations Research 42 (1994) 860-878.

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

[12] M. Garey, D. Johnson, Computers and Intractability: A Guide to Theory of NP-Completeness, W.H. Freeman & Co., New York, 1979.

[13] M. Hifi, A genetic algorithm based heuristic for solving the weighted maximum independent set and some equivalent problems, Journal of the Operational Research Society 48 (6) (1997) 612-622.

[14] D.S. Johnson, M. Trick, Cliques, Coloring, and Satisfiability: Second DIMACS implementation challenge, AMS, Providence, RI, 1996.

[15] G. Karypis, V. Kumar, A fast and high quality multilevel scheme for partitioning irregular graphs, SIAM Journal on Scientific Computing 20 (1998) 359-392.

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

[17] G. Karypis, V. Kumar, Multilevel *k*-way partitioning scheme for irregular graphs, Journal of Parallel and Distributed Computing 48 (1998) 96-129.

[18] A. Mehrotra, M.A. Trick, A column generation approach for graph coloring, INFORMS Journal on Computing 8 (1996) 344-354.

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

[20] G.L. Nemhauser, P.H. Vance, C. Barnhart, E.L. Johnson, Solving binary cutting stock problems by column generation and branch-and-bound, Computational Optimization and Applications 3 (1994) 111-130.

[21] P.M. Pardalos, G.P. Rodgers, A branch and bound algorithm for the maximum clique problem, Computers and Operations Research 19 (5) (1992) 363-375.

[22] C. Ribeiro, M. Minoux, Solving hard constrained shortest path problems by lagrangian relaxation and branch and bound algorithm, Methods of Operations Research 53, in: M. Beckmann et al. (Eds.), Proceedings of the 10th Symposium on Operations Research, Munich, Germany, 1985, pp. 305-316.

[23] M.W.P. Savelsbergh, A branch and price algorithm for the generalized assignment problem, Operations Research 45 (1997) 831-841.

[24] F. Vanderbeck, Decomposition and Column Generation for Integer Programs, Ph.D. Thesis, Universite Catholique de Louvain, Belgium, 1994.

[25] D. Warrier, W.E. Wilhelm, I.V. Hicks, J.S. Warren, A branch and price approach for maximum weight independent set problem, Working Paper, Department of Industrial Engineering, Texas A&M University, 2004.

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

VITA

Sandeep Sachdeva was born on March 3, 1980 in New Delhi, India. He received his Bachelor of Technology degree in mechanical engineering from Indian Institute of Technology, Delhi in May 2002. In Fall 2002, he joined the Industrial Engineering department at Texas A&M University. He received his Master of Science degree in industrial engineering in December 2004.

Permanent Address:

23/3 Shivpuri

Delhi-110051

INDIA

Phone: (91)-11-22416383

E-mail: sandeeps@tamu.edu

ssachdeva@gmail.com