MINIMUM WEIGHTED PERFECT NEIGHBORHOOD SET PROBLEM

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

UMUR HASTÜRK

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
OPERATIONAL RESEARCH

JULY 2020

Approval of the thesis:

MINIMUM WEIGHTED PERFECT NEIGHBORHOOD SET PROBLEM

submitted by **UMUR HASTÜRK** in partial fulfillment of the requirements for the degree of **Master of Science in Operational Research Department, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar
Dean, Graduate School of **Natural and Applied Sciences** _____

Assoc. Prof. Dr. Cem İyigün
Head of Department, **Operational Research** _____

Assist. Prof. Dr. Mustafa Kemal Tural
Supervisor, **Industrial Engineering, METU** _____

Examining Committee Members:

Prof. Dr. Esra Karasakal
Industrial Engineering, METU _____

Assist. Prof. Dr. Mustafa Kemal Tural
Industrial Engineering, METU _____

Prof. Dr. Sinan Gürel
Industrial Engineering, METU _____

Assoc. Prof. Dr. Seçil Savaşaneril Tüfekçi
Industrial Engineering, METU _____

Assoc. Prof. Dr. Ayşegül Altın Kayhan
Industrial Engineering, TOBB ETÜ _____

Date: 20.07.2020

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Surname: Umur Hastürk

Signature :

ABSTRACT

MINIMUM WEIGHTED PERFECT NEIGHBORHOOD SET PROBLEM

Hastürk, Umur
M.S., Department of Operational Research
Supervisor: Assist. Prof. Dr. Mustafa Kemal Tural

July 2020, 118 pages

Given an undirected simple graph $G = (V, E)$, the open neighborhood of a vertex $j \in V$, denoted by $\delta(j)$, is defined as the set of all vertices that are adjacent to j , i.e., $\delta(j) = \{i | \{i, j\} \in E\}$. The closed neighborhood of a vertex j, denoted by $\Delta(j)$, is defined as $\Delta(j) = \delta(j) \cup \{j\}$. For a set $S \subseteq V$, a vertex j is said to be perfect with respect to S, i.e., S-perfect, if $|\Delta(j) \cap S| = 1$. The set S is said to be a perfect neighborhood set if the set of S-perfect vertices dominate G.

Hedetniemi et al. (1997) [1] proposed a linear-time algorithm for the minimum cardinality perfect neighborhood set problem when G is a tree. We observe some flaws in the proposed algorithm and correct them. Moreover, we consider the weighted version of the problem, where the weight of a perfect neighborhood set S is defined as $\sum_{j \in V}(w_j y_j + v_j x_j)$. Here y_j and x_j are binary parameters taking the value 1 if and only if j is in S and j is S-perfect, respectively, and w_j and v_j are the weights associated with vertex j. We extend the algorithm proposed by Hedetniemi et al. for trees to the weighted case and check its correctness by comparing its solutions with the solutions of an integer programming formulation that we propose for arbitrary graphs.

Additionally, we provide some valid inequalities for the integer programming formulation to make it stronger, and characterize the perfect neighborhood set polytope for star graphs and complete graphs by the help of these valid inequalities. Finally, we conduct computational experiments to see the effects of these valid inequalities.

Keywords: Perfect Neighborhood Set, Integer Programming, Valid Inequality, Tree, Dominating Set

ÖZ

MİNİMUM AĞIRLIKLI MÜKEMMEL KOMŞULUK KÜMESİ PROBLEMİ

Hastürk, Umur
Yüksek Lisans, Yöneylem Araştırması Bölümü
Tez Yöneticisi: Dr. Öğr. Üyesi. Mustafa Kemal Tural

Temmuz 2020 , 118 sayfa

Yönsüz basit bir çizge $G = (V, E)$ üzerinde, $\delta(j)$ ile gösterilen $j \in V$ düğümünün açık komşuluk kümesi, j'ye bitişik tüm komşuların kümesi olarak tanımlanır, diğer bir deyişle $\delta(j) = \{i | \{i, j\} \in E\}$. $\Delta(j)$ ile gösterilen j düğümünün kapalı komşuluk kümesi ise $\Delta(j) = \delta(j) \cup \{j\}$ olarak tanımlanır. Bir küme $S \subseteq V$ için, eğer $|\Delta(j) \cap S| = 1$ ise j düğümü S'ye göre mükemmel, diğer bir deyişle S-mükemmel kabul edilir. Eğer S-mükemmel düğümler kümesi, G çizgesinin bir baskınlık kümesi ise, S kümesine mükemmel komşuluk kümesi denir.

Hedetniemi ve diğ. (1997) [1], G bir ağaç olduğunda en az eleman sayılı mükemmel komşuluk kümesi problemini çözen ve doğrusal zamanda çalışan bir algoritma önerdi. Biz, önerilen algoritmada bazı kusurlar gözlemledik ve bu çalışmada bunları düzelttik. Ayrıca, sorunun ağırlıklı versiyonunu ele alarak bir S mükemmel komşuluk kümesinin ağırlığını $\sum_{j \in V}(w_j y_j + v_j x_j)$ ile belirledik. Burada y_j ve x_j, j'nin S'nin içinde ve S-mükemmel olduğu durumlarda 1 değerini alan ikili değişkenlerdir, ve w_j ve v_j, j düğümü ile ilişkilendirilen ağırlık değerleridir. Biz önerilen algoritmayı, ağırlıklı versiyonu ele alacak şekilde genişlettik ve algoritmanın doğruluğunu, her-

hangi bir çizge için önerdiğimiz bir tam sayılı programlama formülasyonundan gelen çözüm değerleri ile kendisinin çözümlerini kıyaslayarak kontrol ettik.

Ayrıca, tam sayılı programlama formülasyonunu daha güçlü hale getirmek için ek geçerli eşitsizlikler sağladık, ve bu geçerli eşitsizliklerin yardımıyla yıldız çizgeler ve tam çizgeler için mükemmel komşuluk kümesi politopunu karakterize ettik. Son olarak, bu geçerli eşitsizliklerin etkilerini görmek için hesaplama deneyleri yaptık.

Anahtar Kelimeler: Mükemmel Komşuluk Kümesi, Tam Sayı Programlama, Geçerli Eşitsizlik, Ağaç, Baskınlık Kümesi

In Loving Memory of Alaska.

ACKNOWLEDGMENTS

I would like to thank my thesis supervisor Assist. Prof. Dr. Mustafa Kemal Tural for his support, guidance, and encouragement through my MSc studies, not only for the thesis but on every aspect of academia.

I would also like to thank the examining committee members, Prof. Dr. Esra Karasakal, Prof. Dr. Sinan Gürel, Assoc. Prof. Dr. Seçil Savaşaneril Tüfekçi, and Assoc. Prof. Dr. Ayşegül Altın Kayhan for their feedback and valuable contributions to enhance the study further.

I would also like to express my sincere thanks to my wife, Nazlı Dolu Hastürk, for her support on the work. She endlessly helped me with her brilliant ideas and always encourages me to pursue the life I dream.

I would also like to thank professors and research assistants of METU IE who contributed to this study directly on indirectly. I would like to express my sincere gratitude to my friends Nur Banu Demir and Deniz Altınpulluk for helping me with C++ and CPLEX.

This study was supported by the Scientific and Technological Research Council of Turkey (TUBITAK).

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

CHAPTER 1

INTRODUCTION

Let $G = (V, E)$ be an undirected simple graph, where V is the set of its vertices and E is the set of its edges. The vertices $i \in V$ and $j \in V$ ($i \neq j$) are said to be adjacent if $\{i, j\} \in E$. The open neighborhood of a vertex j, denoted by $\delta(j)$, is defined as the set of all vertices that are adjacent to j , i.e., $\delta(j) = \{i | \{i, j\} \in E\}$. The closed neighborhood of a vertex j, denoted by $\Delta(j)$, is defined as $\Delta(j) = \delta(j) \cup \{j\}$. The cardinality, i.e., number of elements, of a set A is denoted as $|A|$. The order of a graph $G = (V, E)$ is the cardinality of V. The degree of a vertex j is the number of edges connected to it, which is equal to $|\delta(j)|$. A vertex j is called pendant if $|\delta(j)| = 1$. If a vertex is adjacent to a pendant vertex, then it is named as a support vertex. Therefore, the set of pendant and support vertices are defined as $Pen = \{j | \, |\delta(j)| = 1\}$ and $Sup = \{j | \, \{i, j\} \in E, i \text{ pendant}\}$, respectively. A set $D \subseteq V$ is said to be a dominating set of G if each vertex in $V - D$ is adjacent to at least one vertex in D.

Given a set $S \subseteq V$, a vertex j is said to be perfect with respect to S if $|\Delta(j) \cap S| = 1$. In this case, vertex j is said to be S-perfect. We also denote the set of all S-perfect vertices of G by $S - perfect$. If the set of S-perfect vertices, i.e. $S - perfect$, is a dominating set of G, then S is called a perfect neighborhood set (PN set) of G.

As an example, let $G = (V, E)$ be the graph with $V = \{1, 2, 3\}$ and $E = \{\{1, 2\}$ $, \{2, 3\}\}$ (see Figure 1.1). Note that in this graph, we have $\Delta(1) = \{1, 2\}$, $\Delta(2) = \{1, 2, 3\}$ and $\Delta(3) = \{2, 3\}$.

Let us assume that $S = \{1\}$. In this case, it can be seen that the vertices 1 and 2 are S-perfect as we have $|\Delta(j) \cap S| = 1$ for $j = 1, 2$. This example is displayed in

Figure 1.1: An example graph with three vertices.

Figure 1.2. If a vertex is selected to be in the set S (here the vertex 1), then the vertex is colored in black. Moreover, if a vertex is S-perfect (here the vertices 1 and 2), then a square is placed outside the vertex. It can be observed the set of S-perfect vertices is a dominating set of G (since vertices 1 and 2 are in the set of S-perfect vertices and vertex 3 is adjacent to a vertex in the set of S-perfect vertices, namely vertex 2). Therefore, it can be concluded that the set $S = \{1\}$ is a PN set of G.

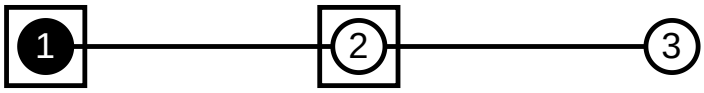


Figure 1.2: Graphical display of the vertices in S and $S - perfect$ when $S = \{1\}$.

For the same graph, now let us assume that $S = \{1, 2\}$ (see Figure 1.3). In this case, we have $|\Delta(1) \cap S| = |\{1, 2\} \cap \{1, 2\}| = 2$, $|\Delta(2) \cap S| = |\{1, 2, 3\} \cap \{1, 2\}| = 2$ and $|\Delta(3) \cap S| = |\{2, 3\} \cap \{1, 2\}| = 1$, implying that the only vertex that is S-perfect is vertex 3. Since the set of S-perfect vertices is not a dominating set of G in this case, we have that $S = \{1, 2\}$ is not a PN set of G.

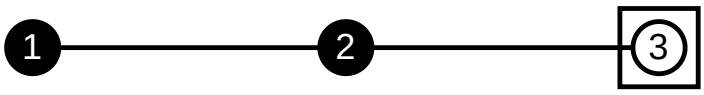


Figure 1.3: Graphical display of the vertices in S and $S - perfect$ when $S = \{1, 2\}$.

Given a graph G, the minimum (maximum) cardinality PN set problem, abbreviated as MinCPNSP (MaxCPNSP), is the problem of finding a PN set of minimum (maximum) cardinality in G. Note that both problems MinCPNSP and MaxCPNSP take into account the cardinalities of PN sets, but not the cardinalities of the sets of perfect

vertices with respect to PN sets. In this thesis, we also consider the cardinalities of the sets of perfect vertices with respect to the PN sets by proposing the minimum weighted perfect neighborhood set problem (MinWPNSP) which aims to find a PN set S in a given graph G of minimum weight, where the weight of a set S is defined as $\sum_{j \in S} w_j + \sum_{j \in S-perfect} v_j$. Here w_j and v_j are the weights associated with the vertex j. When $w_j = 1$ and $v_j = 0$ for all $j \in V$, the MinWPNSP reduces to the MinCPNSP. On the other hand, when $w_j = -1$ and $v_j = 0$ for all $j \in V$, the MinWPNSP reduces to the MaxCPNSP.

In the next chapter, we provide a literature review on perfect neighborhood sets. Then, in Chapter 3, we show that the algorithm proposed by Hedetniemi et al. (1997) [1] to solve the MinCPNSP in trees has some flaws and does not work correctly. In addition to fixing this algorithm, we also extend it to make it work for the weighted problem, i.e., the MinWPNSP, in trees. In Chapter 4, we propose an integer programming (IP) formulation for the MinWPNSP that is applicable to any graph G and in Chapter 5, we compare the IP formulation with our extended algorithm in terms of solution time and check the correctness of the algorithm by comparing the objective function values of the two approaches. In Chapter 6, we provide some valid inequalities for the MinWPNSP and characterize the perfect neighborhood set polytope in star and complete graphs. In Chapter 7, we study the effects of the valid inequalities and finally, we conclude in Chapter 8 with some future research directions.

CHAPTER 2

LITERATURE REVIEW

Perfect neighborhood sets were introduced by Fricke et al. (1999) [2]. The authors defined the lower and upper perfect neighborhood numbers of G, denoted by $\theta(G)$ and $\Theta(G)$, respectively, as the cardinality of the optimal solutions of MinCPNSP and MaxCPNSP, respectively.

A set $S \subseteq V$ of a graph $G = (V, E)$ is called a 2-dominating set if each vertex in $V - S$ is within a distance of at most two edges with at least one vertex of S. A 2-dominating set S is minimal if $S - \{j\}$ is not a 2-dominating set for all $j \in S$. The minimum (maximum) cardinality among all minimal 2-dominating sets of G is denoted by $\gamma_2(G)$ $(\Gamma_2(G))$. It can be shown that every PN set is also a 2-dominating set (see Proposition 6.1 for the proof in our context), and therefore as argued in [2] we have that $\gamma_2(G) \leq \theta(G)$.

Let $\gamma(G)$ $(\Gamma(G))$ denote the minimum (maximum) cardinality among all minimal dominating sets of a graph G. It has been proved in [2] that for any minimal dominating set D, there exists a PN set of G of cardinality $|D|$. This proof gives the immediate results as $\theta(G) \leq \gamma(G)$ and $\Theta(G) \geq \Gamma(G)$. The authors also show that for any graph G, $\Theta(G) \leq \Gamma(G)$ holds true. Overall, the final result is $\Theta(G) = \Gamma(G)$ for an arbitrary graph G.

Closed neighborhood of a set S is defined as the union of the closed neighborhoods of the vertices in S, i.e., $\Delta(S) = \cup_{v \in S} \Delta(v)$. The S-private neighborhood set of a vertex $v \in S$ is defined as $pn[v, S] = \Delta(v) - \Delta(S - \{v\})$. A set S is irredundant if for every vertex $v \in S$, $pn[v, S] \neq \emptyset$. $ir(G)$ $(IR(G))$ represents the lower (upper) irredundance number which is the minimum (maximum) cardinality of a maximal

irredundant set in G. The authors also mentioned a conjecture as for all graphs G, $\theta(G) \leq ir(G)$.

This conjecture has been proved to be true for any tree T by Cockayne et al. (1998) [3]. It is also proved in this paper that if S is a PN set, then S is irredundant. Moreover, the conjecture has also been proved to be true for a graph G if G is claw-free or if G has a maximal irredundant set S of minimum cardinality for which the subgraph induced by S has at most six non-isolated vertices by Cockayne et al. (1999) [4]. However, Favaron and Puech (1999) [5] had a counterexample to this conjecture and proved that the conjecture is not necessarily correct for an arbitrary graph G and $\theta(G) - ir(G)$ can be arbitrarily large.

Hedetniemi et al. (1997) [1] continued to work on $\theta(G)$. They proved that the decision problem of computing $\theta(G)$ (i.e., computing if a graph has a PN set of cardinality at most k) is NP-complete for bipartite and chordal graphs. Additionally, some bounds are provided for $\theta(G)$. The authors combined the result of $\theta(G) \leq \gamma(G)$ with the result that $\gamma(G) \leq n/2$ which holds true for every connected graph G of order $n \geq 2$ (see Haynes et al. (1998) [6]) and showed that if G is a connected graph of order $n \geq 2$, then $\theta(G) \leq n/2$. Moreover, they have also proposed and proved that $\theta(G) \leq n/3$ is satisfied if G is a connected graph of order $n \geq 3$ and if the minimum degree of the vertices is 2. Lastly, they presented a linear time algorithm to compute $\theta(T)$ for any tree T. This algorithm has some flaws and does not work as intended. This is discussed in Chapter 3 in detail.

Later, Xie et al. (2007) [7] worked on the MaxCPNSP. They provided an algorithm that finds $\Theta(T)$ for a given tree T in $\mathcal{O}(n^2)$ time ($n = |V|$).

Denoting $G[S]$ as the subgraph induced by S in $G = (V, E)$, a set S is independent if all vertices of $G[S]$ are isolated, i.e., if the edge set of $G[S]$ is an empty set. Favaron and Puech (1999) [5] have introduced that a set $S \subseteq V$ is called an "Independent perfect neighborhood set" if S is a PN set and if S is an independent set. The authors denote by $\theta_i(G)$ the minimum cardinality of an independent PN set and mention that $\theta(G) \leq \theta_i(G)$ is satisfied. They prove that if G is a claw-free graph, I an independent set of G and $\Delta^2[I] = V$, then I is a PN set of G. Here, $\Delta^2(I)$ represents the double neighborhood, i.e., the set of all vertices of G that can be reached from the set I

within at most two edges. Additionally, Xie et al. (2007) [7] made a conjecture that $\Theta_i(T) = \Theta(T) = IR(T)$, where $\Theta_i(T)$ denotes the maximum cardinality of an independent PN set for a tree T. Favaron (2000) [8], later, mentions that every maximal independent set is a PN set.

A similar definition to PN sets, named as "Open perfect neighborhood sets", is introduced by Hedetniemi (2006) [9]. For these sets, it is still expected that S-perfect vertices are dominating the given graph G, and a vertex j is assumed to be S-perfect if $|\delta(j) \cap S| = 1$. The problems of finding the minimum and maximum cardinality of open perfect neighborhood sets have never been studied in any paper in the literature [10].

The reader should note that in addition to the perfect neighborhood set definition made by Fricke et al. (1999) [2], there has been another definition of perfect neighborhood sets in the literature which is introduced by Sampathkumar and Neeralagi (1994) [11]. The authors define the set $S \subseteq V$ as a PN set if for all $u, v \in S$ $(u \neq v)$, $\Delta(u) \cap \Delta(v) = \emptyset$ is satisfied. There is no direct relationship between these definitions, so one set might be a PN set for only one of the two definitions.

PN sets have direct relations with many concepts in graph theory such as dominating sets, irredundant sets, and packings, see e.g. [1, 2, 7]. Thus, many complicated characteristics of several graph theory problems can be encountered within the problems involving PN sets. Additionally, PN sets can find applications in computer networks, physiology, transportation, etc (see Xie et al. (2007) [7]).

We also observe that the concept is applicable on a security problem on a graph $G = (V, E)$. We place weapons and controllers on the vertices and we assume that a weapon in $i \in V$ should be controlled by a controller in $j \in V$ if $j \in \Delta(i)$. Thus, both the weapon and the controller can be placed on the same vertex. However, neither more than one weapon nor controller can be placed on the same vertex. A controller can control more than one weapon at a time and for a weapon in i, we assume that the control is only possible if there does not exist any other controller in $\Delta(i)$ because of interference. If this is satisfied, we assume that the weapon protects $\Delta(i)$. The aim of the problem is to place weapons and their controllers in a way that all vertices in V are protected. If we also assume the cost of installing weapons or

controllers on the vertices are different than each other, then this problem would be an example of MinWPNSP, where controllers and weapons represent the set S and $S - perfect$, respectively.

Optimization problems related to PN sets in the literature focused on the cardinality of PN sets. Therefore it is assumed that the weights of all vertices are equal to each other. Moreover, none of the studies focused on using IP techniques to solve these optimization problems. Our motivation is to fill these gaps of the literature for the PN sets.

CHAPTER 3

A LINEAR TIME ALGORITHM FOR TREES

In this section, we propose a linear time algorithm for the MinWPNSP in trees. This algorithm generalizes the algorithm proposed by Hedetniemi et al. (1997) [1] for the MinCPNSP in trees. While doing that, we have also changed the authors' algorithm to make it work as intended.

3.1 The Algorithm of Hedetniemi et al. (1997) [1] to Compute $\theta(G)$

Assume that $T_0 = (V, E)$ is a tree with a specified root. For a vertex i, any vertex j of T_0 ($j \neq i$) that lies on the unique path from i to the root of the tree is called an ascendant of i. In this case, we also say that i is a descendant of j. If j is an ascendant of i and $\{i, j\} \in E$, then j is called the parent of i and i is called a child of j. Hedetniemi et al. (1997) [1] assume that the vertex set of the tree T_0 is $V = \{1, 2, 3, \ldots, p\}$, where 1 is the root node of T_0 and for any $i < j$ they have that j is not an ascendant of i. Their algorithm uses an input vector $parent$, called the parent array, where $parent[i]$ represents the parent of i. The algorithm starts with p trees which are all isolated vertices and then joins p to $parent[p]$ by adding an edge between them. Next, it joins $p - 1$ to $parent[p - 1]$ and continues in this manner finally joining 2 to $parent[2] = 1$. Throughout these iterations new trees are formed and after all iterations tree T_0 is constructed. In this thesis, we assume that $p = n = |V|$.

Hedetniemi et al. (1997) [1] start by characterizing all possible (rooted) tree-subset pairs (T, S) which may lead to a PN set of T_0 after the iterations described above are performed. In this notation, T represents a subtree of T_0 which is obtained during the iterations described above and S represents a subset of the vertices of T. Let us

assume that r is the root of T. If none of the conditions below hold true, then the corresponding tree-subset pair cannot yield a PN set of T_0. We call these conditions as Item i and Item ii.

i. If $r \in \Delta(S)$, then every vertex of T, except possibly for r, is S-perfect or adjacent with an S-perfect vertex,

ii. If $r \notin \Delta(S)$, then every vertex of T, except for r and possibly for some neighbors of r, is S-perfect or adjacent with an S-perfect vertex.

Assume that we have two tree-subset pairs, (T_1, S_1) and (T_2, S_2), where r_1 is the root of T_1 and r_2 is the root of T_2. Hedetniemi et al. (1997) [1] combine (T_1, S_1) and (T_2, S_2) by putting an edge between r_1 and r_2, and calling r_1 as the root of the new tree obtained. After this combination, a new tree-subset pair (T, S) is obtained with T having r_1 as its root. This combination operation, denoted by the symbol o, is displayed below

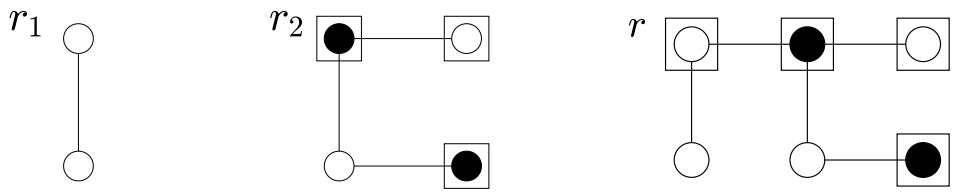
$$(T_1(V_1, E_1), S_1 \subseteq V_1)\ o\ (T_2(V_2, E_2), S_2 \subseteq V_2) = (T(V, E), S \subseteq V)$$

where

$$\begin{aligned} V &= V_1 \cup V_2 \\ E &= E_1 \cup E_2 \cup \{r_1, r_2\} \\ S &= S_1 \cup S_2 \end{aligned}$$

An example of a combination is shown in Figure 3.1. In this example, $S_1 = \emptyset$ and therefore $S = S_2$. In the figure, the vertices in S_1, S_2, and S are colored in black. Note that the root of T which is denoted by r is the same as r_1. Observe that after the combination, the set S obtained is a PN set of T (vertices in squares dominate T).

Then, authors classify possible tree-subset pairs (T, S) (with root r) that may lead to a PN set of T_0 by defining 14 different classes that are listed below.



(a) (T_1, S_1) (b) (T_2, S_2) (c) (T, S)

Figure 3.1: Two tree-subset pairs and their combination.

$[1] = \{(T, S) |\, r \in S$, S is a PN set of T, r is S-perfect, r is adjacent with an S-perfect vertex, r has a neighbor that is not S-perfect and has r as its unique S-perfect neighbor$\}$,

$[2] = \{(T, S) |\, r \in S$, S is a PN set of T, r is S-perfect, r is adjacent with an S-perfect vertex, every neighbor of r is either S-perfect or is adjacent with an S-perfect vertex different from $r\}$,

$[3] = \{(T, S) |\, r \in S$, S is a PN set of T, r is S-perfect, r is not adjacent with an S-perfect vertex, r has a neighbor that has r as its unique S-perfect neighbor$\}$,

$[4] = \{(T, S) |\, r \in S$, S is a PN set of T, r is S-perfect, r is not adjacent with an S-perfect vertex, every neighbor of r is adjacent with an S-perfect vertex different from $r\}$,

$[5] = \{(T, S) |\, r \in S$, S is a PN set of T, r is not S-perfect$\}$,

$[6] = \{(T, S) |\, r \in S$, S is not a PN set of $T\}$,

$[7] = \{(T, S) |\, r \notin S$, S is a PN set of T, r is S-perfect, r is adjacent with an S-perfect vertex, r has a neighbor that is not S-perfect and has r as its unique S-perfect neighbor$\}$,

$[8] = \{(T, S) |\, r \notin S$, S is a PN set of T, r is S-perfect, r is adjacent with an S-perfect vertex, every neighbor of r is either S-perfect or is adjacent with an S-perfect vertex different from $r\}$,

$[9] = \{(T, S) |\ r \notin S$, S is a PN set of T, r is S-perfect, r is not adjacent with an S-perfect vertex, r has a neighbor that has r as its unique S-perfect neighbor$\}$,

$[10] = \{(T, S) |\ r \notin S$, S is a PN set of T, r is S-perfect, r is not adjacent with an S-perfect vertex, every neighbor of r is adjacent with an S-perfect vertex different from $r\}$,

$[11] = \{(T, S) |\ r \notin S$, S is a PN set of T, r is not S-perfect$\}$,

$[12] = \{(T, S) |\ r \notin \Delta(S)$, S is not a PN set of T, every neighbor of r is adjacent with an S-perfect vertex$\}$,

$[13] = \{(T, S) |\ r \notin \Delta(S)$, S is not a PN set of T, r has a neighbor that is neither S-perfect nor is adjacent with an S-perfect vertex$\}$,

$[14] = \{(T, S) |\ r \notin S$, S is not a PN set of T, $|\Delta(r) \cap S| \geq 2\}$.

After the classification of these tree-subset pairs, Hedetniemi et al. create a multiplication table, showing the result of each combination of ordered pairs of 14 classes in Table 3.1. In this table, "$\times$" represents tree-subset pairs that cannot fit into the definitions of Item i or ii. Thus, they are eliminated since they can never form a PN set with any combination.

For example, in Figure 3.1, we have the combination $[13]\ o\ [2] = [7]$ since the tree-subset pairs (T_1, S_1), (T_2, S_2), and (T, S) are in classes $[13]$, $[2]$, and $[7]$, respectively. In the multiplication table, the combination of $[13]$ and $[2]$ (in this order) is claimed to result in the class $[8]$ which is not correct.

The algorithm is initialized with $|V|$ trees, each having a unique vertex of T_0 and no edge. For a tree T consisting of a single vertex i and a class $[\ell]$, a value is assigned and stored in $vector[i, \ell]$ to represent the minimum cardinality of S over all possible subsets S making (T, S) a tree-subset pair that is in class $[\ell]$. Here $vector$ is indeed a matrix with $|V|$ rows and 14 columns. We keep this notation in order not to deviate too much from the notation used in Hedetniemi et al. (1997) [1].

For a tree with a single vertex, say vertex i, we only have two possible tree-subset

Table 3.1: Multiplication table of classes in Hedetniemi et al. (1997) [1].

o	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[12]	[13]	[14]
[1]	×	×	×	×	×	×	×	[1]	×	[1]	[1]	[1]	[1]	[1]
[2]	×	[5]	×	×	[5]	×	×	[2]	×	[1]	[2]	[2]	[2]	[1]
[3]	×	×	×	×	×	×	×	[3]	×	×	[3]	[1]	[1]	[3]
[4]	×	[6]	×	×	[6]	×	×	[4]	×	[3]	[4]	[2]	[2]	[3]
[5]	×	[5]	×	×	[5]	×	×	[5]	×	×	[5]	[5]	[5]	×
[6]	×	[6]	×	×	[6]	×	×	[6]	×	×	[6]	[5]	[6]	×
[7]	×	×	×	×	×	×	[7]	[7]	[7]	[7]	[7]	[7]	[7]	[7]
[8]	[11]	[11]	[11]	[11]	[11]	×	[8]	[8]	[8]	[8]	[8]	[7]	[7]	[7]
[9]	×	×	×	×	×	×	[9]	[9]	[9]	[9]	[9]	[9]	[9]	[9]
[10]	[11]	[11]	×	[11]	[14]	×	[8]	[8]	[8]	[8]	[10]	[9]	[9]	[9]
[11]	[11]	[11]	[11]	[11]	[11]	×	[11]	[11]	[11]	[11]	[11]	×	×	×
[12]	[8]	[8]	[8]	[8]	[10]	[9]	[11]	[11]	[11]	[11]	[12]	[13]	[13]	[13]
[13]	[8]	[8]	[8]	[8]	[10]	[9]	[11]	[11]	[11]	[11]	[13]	[13]	×	[13]
[14]	[11]	[11]	[11]	[11]	[14]	×	[11]	[11]	[11]	[11]	[14]	×	×	×

pairs. One is where we select the vertex to be in S and the other is where $S = \emptyset$. For example, if we define a tree-subset pair (T, S) such as $T = (\{i\}, \emptyset)$ and $S = \{i\}$, then (T, S) would be in class $[4]$ with $|S| = 1$. Otherwise, if $S = \emptyset$, then (T, S) would be in class $[12]$ with $|S| = 0$. Finally, these cardinalities initialize the i^{th} row of $vector$, denoted by $vector[i]$, as $[-, -, -, 1, -, -, -, -, -, -, -, 0, -, -]$, where "$-$" denotes undefined. The steps of the algorithm of Hedetniemi et al. (1997) [1] is shown in Algorithm 1 which starts with this initialization step.

In the line 8 of the Algorithm 1, update procedure of $vector[k]$ is done according to the multiplication table in Table 3.1. For example, $vector[k, 10]$ is updated as follows.

$$vector[k, 10] \coloneqq \min\{vector[k, 10] + vector[j, 11], vector[k, 12] + vector[j, 5], \\ vector[k, 13] + vector[j, 5]\}.$$

In the multiplication table, there are three combinations $[a]\ o\ [b]$ resulting in $[10]$. These are $[10]\ o\ [11] = [10]$, $[12]\ o\ [5] = [10]$, and $[13]\ o\ [5] = [10]$. Thus, in the update of $vector[k, 10]$, among the combinations resulting in class $[10]$, the one with the

Algorithm 1 Algorithm for $\theta(T)$ in Hedetniemi et al. (1997) [1] .

1: **procedure** $\theta(T)$
2: **for** $i \coloneqq 1$ to p **do**
3: initialize $vector[i]$ to $[-,-,-,1,-,-,-,-,-,-,-,0,-,-]$;
4: **end for**
5: **for** $j \coloneqq p$ down to 2 **do**
6: $k \coloneqq parent[j]$;
7: Combine the tree with root j with the tree with root k;
8: Update $vector[k]$ with the combination;
9: **end for**
10: **return** min$\{vector[1,1], vector[1,2], vector[1,3], vector[1,4], vector[1,5],$
11: $vector[1,7], vector[1,8], vector[1,9], vector[1,10], vector[1,11]\}$;
12: **end procedure**

minimum objective function value, i.e., minimum value of $|S|$, is taken into account. Then, the same update applies to $vector[k,\ell]$ for all 14 classes $[\ell]$ in line 8.

Finally, in line 11, authors list all classes with the property that "S is a PN set of T" for the tree-subset pairs (T,S) in which vertex 1 is the root of T. This implies that $T = T_0$. The classes that have this property are [1], [2], [3], [4], [5], [7], [8], [9], [10], and [11]. For each such class $[\ell]$, $vector[1,\ell]$ represents the minimum size of a PN set S in T_0, where (T_0,S) is in class $[\ell]$. Therefore, the minimum of $\{vector[1,1]$, $vector[1,2]$, $vector[1,3]$, $vector[1,4]$, $vector[1,5]$, $vector[1,7]$, $vector[1,8]$, $vector[1,9]$, $vector[1,10]$, $vector[1,11]\}$ gives the cardinality of the minimum cardinality PN set in T_0.

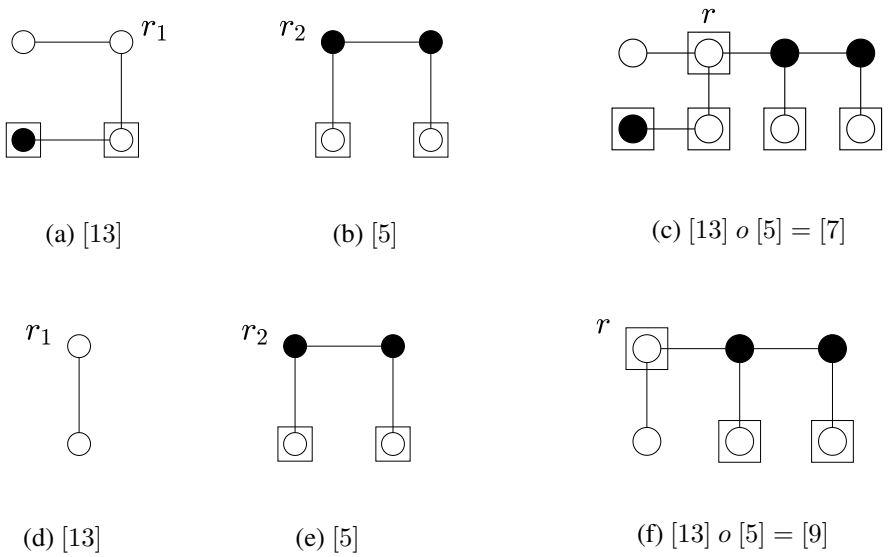
3.2 Corrections on the Algorithm for $\theta(T)$ in Hedetniemi et al. (1997) [1]

As we mentioned before, the algorithm of Hedetniemi et al. (1997) [1] might not work as intended because some of the values in the multiplication table are not correct. Moreover, with the current definitions of the classes, some of the multiplications might result in more than one class, and some classes are unnecessarily split into two subclasses. We now go over these issues.

We assume that two tree-subset pairs (T_1, S_1) and (T_2, S_2) with roots r_1 and r_2, respectively, are combined resulting in the tree-subset pair (T, S) having root $r = r_1$. Moreover, we assume that the tree-subset pairs (T_1, S_1), (T_2, S_2), and (T, S) are of classes $[a]$, $[b]$, and $[c]$, respectively. Therefore, we have that $[a]\ o\ [b] = [c]$. If a combination of two tree-subset pairs does not yield a valid tree-subset pair, then $[c]$ is taken as $\times$.

One of the issues is with the combination $[13]\ o\ [5]$. In the case when r_1 is adjacent to an S_1-perfect vertex, the result would be in class $[7]$, and otherwise it would be in class $[9]$. In the multiplication table given in Table 3.1, it is stated that $[13]\ o\ [5] = [10]$. Examples of combinations $[13]\ o\ [5]$ resulting in $[7]$ and $[9]$ are given in Figure 3.2. We solve this issue by removing the classes $[7]$ and $[9]$, and creating a new class $[7']$ which is the union of $[7]$ and $[9]$. We have checked that defining this new class does not create any problem in the other parts of the multiplication table. The new class $[7']$ is defined below.

$[7'] = \{(T, S) |\ r \notin S$, S is a PN set of T, r is $S-$perfect, r has a neighbor that is not S-perfect and has r as its unique S-perfect neighbor$\}$.



(a) $[13]$ (b) $[5]$ (c) $[13]\ o\ [5] = [7]$

(d) $[13]$ (e) $[5]$ (f) $[13]\ o\ [5] = [9]$

Figure 3.2: Examples to the combination $[13]\ o\ [5]$.

Next, we observe the classes $[1]$ and $[3]$ are unnecessarily split. Therefore we also take their union and create a new class $[1']$ instead of them. This decreases the number of

classes by one as well. The definition of the new class $[1']$ is given below.

$[1'] = \{(T, S) |\ r \in S$, S is a PN set of T, r is S-perfect, r has a neighbor that is not S-perfect and has r as its unique S-perfect neighbor$\}$.

In the new multiplication table, each combination that results in either $[7]$ or $[9]$ is changed into $[7']$ and each combination that results in either $[1]$ or $[3]$ is changed into $[1']$.

An update is also necessary for the class $[11]$, because there might be two possible outcomes of the combination $[11]\ o\ [1']$. Note that this issue exists with $[11]\ o\ [1]$ as well; so it is not a result of redefining class $[1]$. The class $[11]$ has the property that "r is not S-perfect" which implies that $|\Delta(r) \cap S| \neq 1$. In this case, either $|\Delta(r) \cap S| = 0$ or $|\Delta(r) \cap S| \geq 2$. In the former case, r would be S-perfect after the combination $[11]\ o\ [1']$, whereas in the latter one it would continue being not S-perfect. Specifically, we have that $[11]\ o\ [1'] = [8]$ in the former case and $[11]\ o\ [1'] = [11]$ in the latter case. To solve this issue, we split the class $[11]$ into two new classes $[11.1]$ and $[11.2]$ by also storing the information about the cardinality of the set $\Delta(r) \cap S$. The definitions of these new classes are given below.

$[11.1] = \{(T, S) |\ r \notin S$, S is a PN set of T, r is not S-perfect, $|\Delta(r) \cap S| = 0\}$,
$[11.2] = \{(T, S) |\ r \notin S$, S is a PN set of T, r is not S-perfect, $|\Delta(r) \cap S| \geq 2\}$.

After the updates of the classes, the outer part of the multiplication table, i.e., the classes, and a subset of the inner part of the multiplication table, i.e., results of the combinations, are updated in Table 3.2. Examples of tree-subset pairs belonging to each class according to the new class definitions are given in Appendix A.

In Table 3.2, the cells that are grayed out have to be updated. These include the combinations that either result in $[11]$, or one of the classes combined is either $[11.1]$ or $[11.2]$. Now, we will update the grayed out combinations.

Proposition 3.1. $[8]\ o\ [b] = [11.2]$ where $[b]$ is a class with "$r \in S$" and "S is a PN set of T".

Proof. Since we have $r_1 \notin S_1$, it is clear that "$\boldsymbol{r \notin S}$" since $S \supseteq S_1$.

Table 3.2: Updated table with the redefinition of classes.

o	$[1']$	$[2]$	$[4]$	$[5]$	$[6]$	$[7']$	$[8]$	$[10]$	$[11.1]$	$[11.2]$	$[12]$	$[13]$	$[14]$
$[1']$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$[1']$	$[1']$	$[1']$	$[1']$	$[1']$	$[1']$	$[1']$
$[2]$	$\times$	$[5]$	$\times$	$[5]$	$\times$	$\times$	$[2]$	$[1']$	$[2]$	$[2]$	$[2]$	$[2]$	$[1']$
$[4]$	$\times$	$[6]$	$\times$	$[6]$	$\times$	$\times$	$[4]$	$[1']$	$[4]$	$[4]$	$[2]$	$[2]$	$[1']$
$[5]$	$\times$	$[5]$	$\times$	$[5]$	$\times$	$\times$	$[5]$	$\times$	$[5]$	$[5]$	$[5]$	$[5]$	$\times$
$[6]$	$\times$	$[6]$	$\times$	$[6]$	$\times$	$\times$	$[6]$	$\times$	$[6]$	$[6]$	$[5]$	$[6]$	$\times$
$[7']$	$\times$	$\times$	$\times$	$\times$	$\times$	$[7']$	$[7']$	$[7']$	$[7']$	$[7']$	$[7']$	$[7']$	$[7']$
$[8]$	$[11]$	$[11]$	$[11]$	$[11]$	$\times$	$[8]$	$[8]$	$[8]$	$[8]$	$[8]$	$[7']$	$[7']$	$[7']$
$[10]$	$[11]$	$[11]$	$[11]$	$[14]$	$\times$	$[8]$	$[8]$	$[8]$	$[10]$	$[10]$	$[7']$	$[7']$	$[7']$
$[11.1]$	$[11]$	$[11]$	$[11]$	$[11]$	$\times$	$[11]$	$[11]$	$[11]$	$[11]$	$[11]$	$\times$	$\times$	$\times$
$[11.2]$	$[11]$	$[11]$	$[11]$	$[11]$	$\times$	$[11]$	$[11]$	$[11]$	$[11]$	$[11]$	$\times$	$\times$	$\times$
$[12]$	$[8]$	$[8]$	$[8]$	$[10]$	$[7']$	$[11]$	$[11]$	$[11]$	$[12]$	$[12]$	$[13]$	$[13]$	$[13]$
$[13]$	$[8]$	$[8]$	$[8]$	$[10]$	$[7']$	$[11]$	$[11]$	$[11]$	$[13]$	$[13]$	$[13]$	$\times$	$[13]$
$[14]$	$[11]$	$[11]$	$[11]$	$[14]$	$\times$	$[11]$	$[11]$	$[11]$	$[14]$	$[14]$	$\times$	$\times$	$\times$

r_1 is S_1-perfect, having $|\Delta(r_1) \cap S_1| = 1$. With the combination, it is connected with $r_2 \in S$. Thus, we can add that "$\boldsymbol{|\Delta(r) \cap S| = |\Delta(r_1) \cap S_1| + 1 = 2}$", which also implies "$\boldsymbol{r}$ **is not** $\boldsymbol{S}$**-perfect**".

r_1 has lost its perfectness with the combination but since r_1 is adjacent with an S_1-perfect vertex, it is still dominated by the set of S-perfect vertices. Rest of the vertices of T_1 is also dominated by the set of S-perfect vertices since every neighbor of r_1 is either S_1-perfect or is adjacent with an S_1-perfect vertex different from r_1. Additionally, S_2 is a PN set of T_2 and r_2 is connected to $r_1 \notin S$. Therefore, its perfectness will remain the same with the combination and the set of S-perfect vertices would be dominating T_2. Overall, all of the vertices of T are dominated, thus "$\boldsymbol{S}$ **is a PN set of** $\boldsymbol{T}$".

The class that have these properties is $[11.2]$. □

Proposition 3.1 implies the following changes in the Table 3.2:

$$[8]\ o\ [1'] = [11.2]$$

$$[8]\ o\ [2] = [11.2]$$
$$[8]\ o\ [4] = [11.2]$$
$$[8]\ o\ [5] = [11.2]$$

Proposition 3.2. $[10]\ o\ [b] = [11.2]$ where $[b]$ is a class with "$r \in S$", "r is S-perfect", and "S is a PN set of T".

Proof. See the proof Proposition 3.1. The only difference is that, now, r is dominated by r_2 (only), since r_1 is not adjacent with an S_1-perfect vertex. That is the reason we need that $[b]$ has the property of "r is S-perfect". □

Following changes are valid after Proposition 3.2:

$$[10]\ o\ [1'] = [11.2]$$
$$[10]\ o\ [2] = [11.2]$$
$$[10]\ o\ [4] = [11.2]$$

Proposition 3.3. $[14]\ o\ [b] = [11.2]$ where $[b]$ is a class with "r is S-perfect" and "S is a PN set of T".

Proof. Since we have $r_1 \notin S_1$, it is clear that "$\boldsymbol{r \notin S}$".

Moreover, we have $|\Delta(r_1) \cap S_1| \geq 2$. Since $S \supseteq S_1$, we add that "$\boldsymbol{|\Delta(r) \cap S| \geq |\Delta(r_1) \cap S_1| \geq 2}$". This also implies "**$r$ is not S-perfect**".

S_2 is a PN set for T_2 and r_2 is connected with $r_1 \notin S_1$ with the combination. Thus, r_2 remains to be perfect and then it can be observed that the set of S-perfect vertices dominates T_2. We also know that S_1 is not a PN set for T_1 because r_1 is not dominated by the set of S_1-perfect vertices. After the combination, it is dominated by r_2 and T_1 gets dominated by the set of S-perfect vertices as well. Thus, "**S is a PN set of T**".

The class that satisfies those properties is $[11.2]$. □

With the Proposition 3.3, the following changes occurred on Table 3.2:

$$[14]\ o\ [1'] = [11.2]$$
$$[14]\ o\ [2] = [11.2]$$
$$[14]\ o\ [4] = [11.2]$$
$$[14]\ o\ [7'] = [11.2]$$
$$[14]\ o\ [8] = [11.2]$$
$$[14]\ o\ [10] = [11.2]$$

Proposition 3.4. $[13]\ o\ [b] = [13]$ where $[b]$ is a class with "$r \notin S$" and "S is a PN set of T".

Proof. Since we have $r_1 \notin \Delta(S_1)$, $\Delta(r) = \Delta(r_1) \cup \{r_2\}$ and $r_2 \notin S_2$, it is clear that "$\boldsymbol{r \notin \Delta(S)}$".

While combining two trees where both of the roots are not in S_1 and S_2, respectively, we can say that $S - perfect = S_1 - perfect \cup S_2 - perfect$. Since we know S_2 is a PN set of T_2, we can add that T_2 is dominated by the set of S-perfect vertices as well.

Therefore, since $|\Delta(r_1) \cap S_1| = |\Delta(r) \cap S| = 0$, we can add that both of the properties of $[13]$ as "**S is not a PN set of T**" and "**r has a neighbor that is neither S-perfect nor is adjacent with an S-perfect vertex**" remain the same. Moreover, resulting class would not be $\times$ since we have shown that all of the vertices that are outside of the $\Delta(r)$ are dominated (note that T_2 is dominated by the set of S-perfect vertices).

All of the properties of $[13]$ remain the same with this combination, thus we can conclude that $[13]\ o\ [b] = [13]$. □

By the proof of Proposition 3.4, the following values are updated on the Table 3.2:

$$[13]\ o\ [7'] = [13]$$
$$[13]\ o\ [8] = [13]$$
$$[13]\ o\ [10] = [13]$$

Note that the values we have changed here were initially $[11]$ and now they are neither $[11.1]$ nor $[11.2]$. This is because the initial value of $[11]$ were mistaken.

Proposition 3.5. $[12]\ o\ [b] = [11.1]$ where $[b]$ is a class with "$r \notin S$", "r is S-perfect", and "S is a PN set of T".

Proof. Since we have $r_1 \notin S_1$, it is clear that "$\boldsymbol{r \notin S}$".

With the combination, r_1 is connected to $r_2 \notin S_2$, which does not change the perfectness of r_1. Thus, "**$\boldsymbol{r}$ is not $\boldsymbol{S}$-perfect**".

Moreover, since r_2 is S_2-perfect and $r_1 \notin S_1$, r_2 will remain to be perfect after the combination and thus, the set of S-perfect vertices will dominate T_2. Then, r_2 will dominate r and the remaining vertices of the T_1 is already dominated by the set of S_1-perfect vertices. Overall, "**$\boldsymbol{S}$ is a PN set of $\boldsymbol{T}$**".

$[12]$ claims $r_1 \notin \Delta(S_1)$ and we have $\Delta(r) = \Delta(r_1) \cup \{r_2\}$. Since $r_2 \notin S$, last claim would be "$\boldsymbol{|\Delta(r) \cap S| = 0}$".

These properties are satisfied by class $[11.1]$. □

The following values are changed on Table 3.2 after Proposition 3.5:

$$[12]\ o\ [7'] = [11.1]$$
$$[12]\ o\ [8] = [11.1]$$
$$[12]\ o\ [10] = [11.1]$$

Proposition 3.6. $[11.1]\ o\ [b] = [8]$ where $[b]$ is a class with "$r \in S$" and "S is a PN set of T".

Proof. Since we have $r_1 \notin S_1$, it is clear that "$\boldsymbol{r \notin S}$".

We have $|\Delta(r_1) \cap S_1| = 0$. With the combination, it is connected with $r_2 \in S$. Thus, we can add that "$|\Delta(r) \cap S| = |\Delta(r_1) \cap S_1| + 1 = 1$", which implies "**$\boldsymbol{r}$ is $\boldsymbol{S}$-perfect**".

Before the combination, S_1 is a PN set of T_1 and $r_1 \notin \Delta(S_1)$. Therefore, there exists at least one vertex, say $u \in \delta(r_1)$, which is S_1-perfect and dominates r_1, and remains

to be S-perfect with the combination. Since $u \in \Delta(r)$, we add that "$\boldsymbol{r}$ **is adjacent with an** $\boldsymbol{S}$**-perfect vertex**".

Moreover, $r_1 \notin S_1$, the perfectness of r_2 does not change with the combination. Thus, $S - perfect = S_1 - perfect \cup S_2 - perfect \cup \{r\}$. Since, both the sets of S_1-perfect and S_2-perfect vertices dominate T_1 and T_2, respectively, it is implied that "$\boldsymbol{S}$ **is a PN set of** $\boldsymbol{T}$". Additionally, $S_1 - perfect \cup S_2 - perfect$ defines a PN set of T without r, thus we have "**every neighbor of** $\boldsymbol{r}$ **is either** $\boldsymbol{S}$**-perfect or is adjacent with an** $\boldsymbol{S}$**-perfect vertex different from** $\boldsymbol{r}$".

The resulting class of the combination, then, would be $[8]$. □

Proposition 3.6 implies the following changes on the Table 3.2:

$$[11.1]\ o\ [1'] = [8]$$
$$[11.1]\ o\ [2] = [8]$$
$$[11.1]\ o\ [4] = [8]$$
$$[11.1]\ o\ [5] = [8]$$

Proposition 3.7. $[11.1]\ o\ [b] = [11.1]$ where $[b]$ is a class with "$r \notin S$" and "S is a PN set of T".

Proof. Since we have $r_1 \notin S_1$, it is clear that "$\boldsymbol{r \notin S}$".

We have $|\Delta(r_1) \cap S_1| = 0$. With the combination, r_1 is connected with $r_2 \notin S$. Thus, we can add that "$\boldsymbol{|\Delta(r) \cap S| = |\Delta(r_1) \cap S_1| = 0}$", which also implies "$\boldsymbol{r}$ **is not** $\boldsymbol{S}$**-perfect**".

Additionally, we know that $r_1 \notin S_1$ and $r_2 \notin S_2$. Thus, none of the perfectness of the vertices are changed with the combination, $S - perfect = S_1 - perfect \cup S_2 - perfect$. Since, both the sets of S_1-perfect and S_2-perfect vertices dominate T_1 and T_2, respectively, it is implied that "$\boldsymbol{S}$ **is a PN set of** $\boldsymbol{T}$".

All of these properties are satisfied with $[11.1]$. □

By the Proposition 3.7, following combinations are changed on Table 3.2:

$$[11.1]\ o\ [7'] = [11.1]$$
$$[11.1]\ o\ [8] = [11.1]$$
$$[11.1]\ o\ [10] = [11.1]$$
$$[11.1]\ o\ [11.1] = [11.1]$$
$$[11.1]\ o\ [11.2] = [11.1]$$

Proposition 3.8. $[a]\ o\ [6] = [7']$ where $[a]$ is a class with "$r \notin \Delta(S)$".

Proof. Note that "$r \notin \Delta(S)$" is equivalent of $|\Delta(r) \cap S| = 0$.

Since we have $r_1 \notin \Delta(S_1)$, it is clear that "$\boldsymbol{r \notin S}$".

r_1 is connected with $r_2 \in S_2$ with the combination. Thus, $|\Delta(r) \cap S| = |\Delta(r_1) \cap S_1| + 1 = 1$, implying "**$\boldsymbol{r}$ is $\boldsymbol{S}$-perfect**".

In $[6]$, every vertex except r_2 is dominated by the set of S_2-perfect vertices. With the combination, since r is S-perfect, r_2 would also be dominated by r. This results that the set of S-perfect vertices dominates T_2. Then, $[a]$ is a class with $r \notin \Delta(S)$. We know by Item ii that the set of S_1 of such a class would be a PN set if we assume that the root, r_1, is also perfect. It is already implied that r_1 gets perfect with the combination. Thus, we add that the set of S-perfect vertices dominates T_1 as well. Overall, "**$\boldsymbol{S}$ is a PN set of $\boldsymbol{T}$**".

Since r_2 is not dominated by the set of S_2-perfect vertices and is dominated by the set of S-perfect vertices, it is implied that r_2 is only dominated by r. Thus, "**$\boldsymbol{r}$ has a neighbor that is not $\boldsymbol{S}$-perfect and has $\boldsymbol{r}$ as its unique $\boldsymbol{S}$-perfect neighbor**".

The class satisfies all properties would be $[7']$. □

Proposition 3.8 enables us to change the following combination:

$$[11.1]\ o\ [6] = [7']$$

Proposition 3.9. $[a]\ o\ [b] = [13]$ where $[a]$ is a class with "$r \notin \Delta(S)$", and $[b]$ is a class with "$r \notin S$", "S is not a PN set of T", and "every neighbor of r is adjacent

with an S-perfect vertex".

Proof. Note that $[a]$ includes the classes $[11.1]$, $[12]$ and $[13]$. Moreover, $[b]$ includes $[12]$ and $[14]$. "Every neighbor of r is adjacent with an S-perfect vertex" is not mentioned in class $[14]$, but if it is not true, $[14]$ has no chance of returning a PN set with any combination and then will not fit into the definition of Item ii. Thus, this is a valid information for $[14]$.

Since we have $r_1 \notin \Delta(S_1)$, $\Delta(r) = \Delta(r_1) \cup \{r_2\}$ and $r_2 \notin S_2$, it is clear that "$\boldsymbol{r \notin \Delta(S)}$", which also implies that r is not S-perfect.

Additionally, every neighbor of r_2 is adjacent with an S_2-perfect vertex and S_2 is not a PN set of T_2. Thus, r_2 is not dominated by the set of S_2-perfect vertices. By the combination with $[a]$, it is connected with $r \notin S$, which is not an S-perfect vertex. Thus, r_2 is also not dominated by the set of S-perfect vertices. Thus, "**$\boldsymbol{r}$ has a neighbor that is neither $\boldsymbol{S}$-perfect nor is adjacent with an $\boldsymbol{S}$-perfect vertex**", which also implies that "**$\boldsymbol{S}$ is not a PN set of $\boldsymbol{T}$**".

By the definition in Item ii, we know that all the vertices such that $u \notin \Delta(r_1)$ are dominated by the set of S_1-perfect vertices. This is again valid with the combination, where all the vertices such that $u \notin \Delta(r)$ are dominated by the set of S-perfect vertices since every neighbor of r_2 is adjacent with an S_2-perfect vertex. Overall, the combined class would still be in the definition of Item ii and would not be classified as $\times$.

These properties are, then, satisfied by class $[13]$ only. □

The following changes on Table 3.2 are done with the Proposition 3.9:

$$[11.1]\ o\ [12] = [13]$$
$$[11.1]\ o\ [14] = [13]$$

Proposition 3.10. $[a]\ o\ [13] = \times$ where $[a]$ is a class with "$r \notin S$".

Proof. In $[13]$, we have at least one vertex, say $u \in \delta(r_2)$, that is neither S_2-perfect nor is adjacent with an S_2-perfect vertex. Additionally, it can be observed that $|\Delta(r_2) \cap$

$S_2| = 0$ (see Item ii). With the combination of $[a]\ o\ [13]$, we add an edge between r_2 and $r \notin S$, implying that $|\Delta(r_2) \cap S| = |\Delta(r_2) \cap S_2| + 0 = 0$. Thus, r_2 would still not be perfect and u will not be dominated by the set of S-perfect vertices. After the combination of $[a]\ o\ [13]$, distance between r and u would be 2, implying $u \notin \Delta(r)$. A tree-subset pair would not fit into Item i or ii if it has at least one vertex that is neither S-perfect nor is adjacent with an S-perfect vertex outside of $\Delta(r)$. Therefore, all of these combination will result "$\times$". □

Proposition 3.10 gives the proof of $[11.1]\ o\ [13] = \times$ and $[11.2]\ o\ [13] = \times$. Additionally, following list of changes have made in Table 3.2 with the Proposition 3.10:

$$[7']\ o\ [13] = \times$$
$$[8]\ o\ [13] = \times$$
$$[10]\ o\ [13] = \times$$
$$[12]\ o\ [13] = \times$$

Proposition 3.11. $[11.2]\ o\ [b] = [11.2]$ where $[b]$ is a class with "S is a PN set of T".

Proof. Since $r_1 \notin S_1$, it is clear that "$\boldsymbol{r \notin S}$".

We know that $|\Delta(r_1) \cap S_1| \geq 2$ and it is clear that $|\Delta(r) \cap S| \geq |\Delta(r_1) \cap S_1|$. Thus, it is implied that "$\boldsymbol{r}$ **is not** $\boldsymbol{S}$**-perfect**" and "$\boldsymbol{|\Delta(r) \cap S| \geq 2}$".

Additionally, none of the perfectness of the vertices changes with the combination. We have already shown that r_1 will remain to be not perfect. r_2, on the other hand, is connected with $r_1 \notin S$, which implies that the perfectness of r_2 will not change as well. Overall, since both $[11.2]$ and $[b]$ are classes with the property of "S is a PN set of T", it is implied that the sets of S_1-perfect and S_2-perfect vertices dominate T_1 and T_2, respectively. We, then, conclude that "$\boldsymbol{S}$ **is a PN set of** $\boldsymbol{T}$".

These results are combined on the class $[11.2]$. □

After Proposition 3.11, we change the following combinations on Table 3.2:

$$[11.2]\ o\ [1'] = [11.2]$$

$$[11.2]\ o\ [2] = [11.2]$$
$$[11.2]\ o\ [4] = [11.2]$$
$$[11.2]\ o\ [5] = [11.2]$$
$$[11.2]\ o\ [7'] = [11.2]$$
$$[11.2]\ o\ [8] = [11.2]$$
$$[11.2]\ o\ [10] = [11.2]$$
$$[11.2]\ o\ [11.1] = [11.2]$$
$$[11.2]\ o\ [11.2] = [11.2]$$

Proposition 3.12. $[11.2]\ o\ [b] = \times$ where $[b]$ is a class with "S is not a PN set of T".

Proof. We know that $|\Delta(r_1) \cap S_1| \geq 2$ and it is clear that $|\Delta(r) \cap S| \geq |\Delta(r_1) \cap S_1|$. Thus, it is implied that "r is not S-perfect", which also means r will never be perfect by any combination because $|\Delta(r) \cap S|$ never decreases. Additionally, r_2 is connected with $r_1 \notin S_1$, which implies that the perfectness of r_2 will not change by this combination as well.

Overall, we have $S - perfect = S_1 - perfect \cup S_2 - perfect$. Since S_2 is not a PN set of T_2, there exists at least one vertex, say $u_2 \in V_2$, which is not dominated by the set of S-perfect vertices will never be dominated by any combination that is made by edge connections from r. Therefore, the result of the combined tree-subset pair would be "$\times$". □

None of the values of the Table 3.2 are changed with the Proposition 3.12, but we mentioned about it since we are covering all combinations that are grayed out on the table.

Proposition 3.13. $[a]\ o\ [11.1] = [a]$ where $[a]$ is a class with "$r \in S$" and that has not the property of "r is not adjacent with an S-perfect vertex".

Proof. Note that there are three classes that satisfy the properties listed for $[a]$; $[1']$, $[2]$ and $[5]$. $[4]$ is not valid since it has the property of "r is not adjacent with an S-perfect vertex". Even though it is not explicitly written, same applies for $[6]$ because

otherwise, $[6]$ would be a class with the property "S is a PN set of T". Overall, in all classes of $[a]$, we have "S is a PN set of T".

With such a combination, all of the properties of class $[a]$ remains the same. First of all, $r_1 \in S_1$ implies $r \in S$.

Next, since $r_1 \in S_1$ is connected to $r_2 \notin S_2$ with the combination, the perfectness of r_1 is not changed. r_2 gets, on the other hand, perfect with the combination and we can imply the set of S-perfect (or S_2-perfect) vertices dominates T_2. In such a case S would be PN set for T, since S_1 is a PN set of T_1.

If every neighbor of r_1 is S_1-perfect or is adjacent with an S_1-perfect vertex different from r_1, this property will remain the same with the combination since $\Delta(r) = \Delta(r_1) \cup \{r_2\}$, and r_2 becomes an S-perfect vertex with the combination. Else, there exist $u \in \delta(r_1)$ which is only dominated by r_1 before the combination, implying u will only be dominated by r since none of the perfectness of the vertices changes after the combination. Therefore, this property also remains to be the same with the combination.

To sum up, all of the properties are valid after the combination. □

By the Proposition 3.13, we do not change any value in Table 3.2 but give the proof of combinations $[1'] \; o \; [11.1] = [1']$, $[2] \; o \; [11.1] = [2]$, and $[5] \; o \; [11.1] = [5]$.

Proposition 3.14. $[4] \; o \; [11.1] = [2]$ and $[6] \; o \; [11.1] = [5]$.

Proof. See Proposition 3.13. The only different is that this time r_1 is not adjacent with an S_1-perfect vertex. After the combination, r would be adjacent with S-perfect vertex, r_2, and the combination class will not have this property. Instead, it will have that "r is adjacent with an S-perfect vertex". For the class $[4]$ ($[6]$), if we make r to be adjacent with an S-perfect vertex, we will end up having the class of $[2]$ ($[5]$). □

Both of the combinations in Proposition 3.14 are changed on Table 3.2.

Proposition 3.15. $[a] \; o \; [11.1] = [a]$ where $[a]$ is a class with "$r \notin S$".

Proof. None of the properties of $[a]$ changes with such a combination, similar to the ones that are explained in Proposition 3.13. First of all, $r_1 \notin S_1$ implies $r \notin S$. Additionally, if $r_1 \notin \Delta(S_1)$, then we have $r \notin \Delta(S)$ since $r_2 \notin S_2$.

Since $r_1 \notin S_1$ and $r_2 \notin S_2$, we have $S - perfect = S_1 - perfect \cup S_2 - perfect$. Thus, S would (not) be a PN set for T, if and only if S_1 is (not) a PN set of T_1. Additionally, r_1 is connected to $r_2 \notin S$, implying that the perfectness of r_1 will not change as well, which also implies $|\Delta(r) \cap S| = |\Delta(r_1) \cap S_1|$.

Finally, unlike in the situation of Proposition 3.13, r_2 is now connected with $r_1 \notin S_1$. Thus, the perfectness of r_2 does not change with the combination. This adds that the information of "r_1 is (not) adjacent with a S_1-perfect vertex" does not change with the connection to r_2 as well.

Overall, all properties of $[a]$ are still satisfied by the combination. □

With Proposition 3.15, all values remain to be same in the Table 3.2.

Proposition 3.16. $[a]\ o\ [11.2] = [a]$ for all classes $[a]$.

Proof. See Proposition 3.13 and 3.15.

The difference is that, now, $r_1 \in S_1$ might be valid for some classes. If $r_1 \in S_1$, then $r \in S$, so the property remains.

Additionally, the set of S_2-perfect vertices will always dominate T_2 with any combination since S_2 is a PN set of T_2 and r_2 is not an S_2-perfect vertex. Then, S would (not) be a PN set for T, if and only if S_1 is (not) a PN set of T_1.

r_2 will have not any chance to be perfect with any combination since $|\Delta(r_2) \cap S_2| \geq 2$. Because the cardinality of this set never decreases, the property of "r_1 is (not) adjacent with a S_1-perfect vertex" will not be affected by the combination for all classes $[a]$.

□

Note that this proposition does not change any value in the Table 3.2. Moreover, with the Proposition 3.16, we covered all combinations that are grayed out in Table 3.2.

However, before finalizing our table, some of the initial values of some combinations are changed to make the function work as intended. Now, we will go over those changes.

Proposition 3.17. $[6]\ o\ [13] = [5]$.

Proof. $r_1 \in S_1$ and $r = r_1$ imply that "$\boldsymbol{r \in S}$". Additionally, since $r_1 \in \Delta(S_1)$ and S_1 is not a PN set of T_1, we have that r_1 is neither S_1-perfect nor is adjacent to an S_1-perfect vertex (see Item i). Thus, we have that $|\Delta(r_1) \cap S_1| \neq 1$, implying that $|\Delta(r) \cap S| = |\Delta(r_1) \cap S_1| \neq 1$ since $r_2 \notin S$, which means that "**r is not S-perfect**".

Next, we have that $r_2 \notin \Delta(S_2)$, implying that $|\Delta(r_2) \cap S_2| = 0$. After the combination, $\Delta(r_2)$ is updated to $\Delta(r_2) \cup \{r_1\}$. With $r_1 \in S$, we have that $|\Delta(r_2) \cap S| = |\Delta(r_2) \cap S_2| + 1 = 1$, implying that r_2 is S-perfect. With the definition of Item ii, it is known that the vertices in T_2 are all dominated by the set of S-perfect vertices when r_2 is S-perfect. In this case, r_2 dominates r as well, and we have that the remaining vertices of T_1 are dominated by the set of S_1-perfect vertices (see Item i), implying that "**S is a PN set of T**".

By combining these results, it is observed that (T, S) would be in class $[5]$. □

Proposition 3.18. $[13]\ o\ [b] = [7']$ where $[b]$ is a class with "$r \in S$".

Proof. $r_1 \notin \Delta(S_1)$ and $r = r_1$ imply that "$\boldsymbol{r \notin S}$". Next, we have that $|\Delta(r_1) \cap S_1| = 0$. r_1 gets connected to $r_2 \in S_2$ after the combination. Thus, $|\Delta(r) \cap S| = |\Delta(r_1) \cap S_1| + 1 = 1$, implying that "**$r$ is S-perfect**".

Item i implies that for a class with $r \in \Delta(S)$, every vertex except possibly the root, is dominated by the set of S-perfect vertices. Since r is S-perfect, r_2 is dominated by the set of S-perfect vertices after the combination, implying that all the vertices of T_2 are dominated by the set of S-perfect vertices in the combination. Additionally, the vertices of T_1 are also dominated by the set of S-perfect vertices in the combination since r is S-perfect (see Item ii). Overall, "**S is a PN set of T**". r_1 has a neighbor that is neither S_1-perfect nor is adjacent to an S_1-perfect vertex. Thus "**r has a neighbor that is not S-perfect and has r as its unique S-perfect neighbor**". The class with these properties is $[7']$. □

Proposition 3.18 enables us to make the following changes on the Table 3.2:

$$[13]\ o\ [1'] = [7']$$
$$[13]\ o\ [2] = [7']$$
$$[13]\ o\ [4] = [7']$$
$$[13]\ o\ [5] = [7']$$

After Proposition 3.18, we finalized our updated table which can be found in Table 3.3.

Table 3.3: Final version of the multiplication table.

o	$[1']$	$[2]$	$[4]$	$[5]$	$[6]$	$[7']$	$[8]$	$[10]$	$[11.1]$	$[11.2]$	$[12]$	$[13]$	$[14]$
$[1']$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$[1']$	$[1']$	$[1']$	$[1']$	$[1']$	$[1']$	$[1']$
$[2]$	$\times$	$[5]$	$\times$	$[5]$	$\times$	$\times$	$[2]$	$[1']$	$[2]$	$[2]$	$[2]$	$[2]$	$[1']$
$[4]$	$\times$	$[6]$	$\times$	$[6]$	$\times$	$\times$	$[4]$	$[1']$	$[2]$	$[4]$	$[2]$	$[2]$	$[1']$
$[5]$	$\times$	$[5]$	$\times$	$[5]$	$\times$	$\times$	$[5]$	$\times$	$[5]$	$[5]$	$[5]$	$[5]$	$\times$
$[6]$	$\times$	$[6]$	$\times$	$[6]$	$\times$	$\times$	$[6]$	$\times$	$[5]$	$[6]$	$[5]$	$[5]$	$\times$
$[7']$	$\times$	$\times$	$\times$	$\times$	$\times$	$[7']$	$[7']$	$[7']$	$[7']$	$[7']$	$[7']$	$\times$	$[7']$
$[8]$	$[11.2]$	$[11.2]$	$[11.2]$	$[11.2]$	$\times$	$[8]$	$[8]$	$[8]$	$[8]$	$[8]$	$[7']$	$\times$	$[7']$
$[10]$	$[11.2]$	$[11.2]$	$[11.2]$	$[14]$	$\times$	$[8]$	$[8]$	$[8]$	$[10]$	$[10]$	$[7']$	$\times$	$[7']$
$[11.1]$	$[8]$	$[8]$	$[8]$	$[8]$	$[7']$	$[11.1]$	$[11.1]$	$[11.1]$	$[11.1]$	$[11.1]$	$[13]$	$\times$	$[13]$
$[11.2]$	$[11.2]$	$[11.2]$	$[11.2]$	$[11.2]$	$\times$	$[11.2]$	$[11.2]$	$[11.2]$	$[11.2]$	$[11.2]$	$\times$	$\times$	$\times$
$[12]$	$[8]$	$[8]$	$[8]$	$[10]$	$[7']$	$[11.1]$	$[11.1]$	$[11.1]$	$[12]$	$[12]$	$[13]$	$\times$	$[13]$
$[13]$	$[7']$	$[7']$	$[7']$	$[7']$	$[7']$	$[13]$	$[13]$	$[13]$	$[13]$	$[13]$	$[13]$	$\times$	$[13]$
$[14]$	$[11.2]$	$[11.2]$	$[11.2]$	$[14]$	$\times$	$[11.2]$	$[11.2]$	$[11.2]$	$[14]$	$[14]$	$\times$	$\times$	$\times$

Since the multiplication table has changed, we have also made a couple of changes in Algorithm 1. The only change in the algorithm is indeed due to the changes in the multiplication table. The steps and the running time of the updated algorithm are the same with those of the original one.

First change is on line 3 in Algorithm 1, where we initialized $vector$ for two classes only ($[4]$ and $[12]$) for each one vertex tree-subset pairs. After the index of the classes changes with the update, we changed line 3 to

$$\text{initialize } vector[i] \text{ to } [-,-,1,-,-,-,-,-,-,-,0,-,-];.$$

Here the ℓ^{th} entry of $vector[i]$ corresponds to the ℓ^{th} class in the new multiplication table. Next, update procedure is now done according to the new table. For example, $vector[k, 10]$ is now updated as

$$vector[k, 10] \coloneqq \min\{vector[k, 10] + vector[j, 11.1], vector[k, 10] + vector[j, 11.2], \\ vector[k, 12] + vector[j, 5]\}.$$

Finally, in line 11, we list all the classes with the property that "S is a PN set of T" for the tree-subset pairs in which vertex 1 is the root. With the current table, the list of the classes that have the property would be $[1']$, $[2]$, $[4]$, $[5]$, $[7']$, $[8]$, $[10]$, $[11.1]$, and $[11.2]$. Thus, the corresponding line is changed into

$$\textbf{return } \min\{vector[1, 1'], vector[1, 2], vector[1, 4], vector[1, 5], vector[1, 7'], \\ vector[1, 8], vector[1, 10], vector[1, 11.1], vector[1, 11.2]\};.$$

Algorithm 2 is the final version of the algorithm for $\theta(T)$ in which all of the modifications listed above are implemented.

Algorithm 2 Updated version of the algorithm for $\theta(T)$.

1: **procedure** $\theta(T)$
2: **for** $i \coloneqq 1$ to n **do**
3: initialize $vector[i]$ to $[-, -, 1, -, -, -, -, -, -, -, 0, -, -]$;
4: **end for**
5: **for** $j \coloneqq n$ down to 2 **do**
6: $k \coloneqq parent[j]$;
7: Combine the tree with root j with the tree with root k
8: Update $vector[k]$ with the combination;
9: **end for**
10: **return** $\min\{vector[1, 1'], vector[1, 2], vector[1, 4], vector[1, 5], vector[1, 7'],$
11: $vector[1, 8], vector[1, 10], vector[1, 11.1], vector[1, 11.2]\}$;
12: **end procedure**

3.3 Generalization of the Algorithm for the Weighted Cardinality

In this section, we extend Algorithm 2 to the weighted case in order to solve the MinWPNSP. In the MinWPNSP, we are given a tree $T_0 = (V, E)$ and two weights w_j and v_j for each vertex $j \in V$ and we aim to find a PN set S in T_0 of minimum weight, where the weight of a set S is defined as $\sum_{j \in S} w_j + \sum_{j \in S-perfect} v_j$. Here w_j and v_j are the weights associated with the vertex j. The weight of selecting the vertex in the set S and $S - perfect$ would be w_j and v_j, respectively.

The extended algorithm will be very similar to Algorithm 2 because the same multiplication table, i.e., Table 3.3, is used since the results of the combinations are not related with the weights. The only changes will be in the initialization step, i.e., line 3 of Algorithm 2, and in the update of $vector[k]$ in line 8 of Algorithm 2.

First, we modify the initialization step of the algorithm. For a tree with a single vertex, say vertex i, we only have two possible tree-subset pairs. In the first one, we select the vertex to be in S and in the second one $S = \emptyset$. For example, if we define a tree-subset pair (T, S) such as $T = (\{i\}, \emptyset)$ and $S = \{i\}$, then (T, S) would be in class $[4]$ with the set S having weight $w_i + v_i$. Otherwise, if $S = \emptyset$, then (T, S) would be in class $[12]$ with S having weight 0. Finally, these weights initialize the i^{th} row of $vector$, denoted by $vector[i]$, as $[-, -, w_i + v_i, -, -, -, -, -, -, -, 0, -, -]$.

Next, we modify the update step in line 8 of Algorithm 2. When two tree-subset pairs (T_1, S_1) and (T_2, S_2) are combined to give us another tree-subset pair (T, S), we know that $S = S_1 \cup S_2$. However, the weight of S does not need to be equal to the sum of the weights of S_1 and S_2. This is because S-perfect is not necessarily equal to $S_1 - perfect \cup S_2 - perfect$. However, it can be shown that $S - perfect/\{r_1, r_2\} = (S_1 - perfect/\{r_1\}) \cup (S_2 - perfect/\{r_2\})$, where r_1 and r_2 are the roots of T_1 and T_2, respectively. Therefore, when computing the weight of S, we have to give special attention to r_1 and r_2. There are 4 cases to consider with respect to r_1. If r_1 is S_1-perfect (before the combination) and S-perfect (after the combination) or if r_1 is neither S_1-perfect nor S-perfect, then the weight of S_1 and its contribution to the weight of S are the same. On the other hand, if r_1 is S_1-perfect (before the combination), but not S-perfect (after the combination), then the contribution of S_1

to the weight of S becomes the weight of S_1 minus v_{r_1}. Similarly, if r_1 is S-perfect, but not S_1-perfect, then the contribution of S_1 to the weight of S becomes the weight of S_1 plus v_{r_1}. For this purpose, we create Table 3.4, which is a matrix named per, that stores what happens to the perfectness of the root of T_1 when (T_1, S_1) that is in class $[i]$ and (T_2, S_2) that is in class $[j]$ are combined. For example, the table shows that if (T_1, S_1) is in class $[1']$ and (T_2, S_2) is in class $[2]$, then r_1 was perfect before the combination and became non-perfect after the combination. The value of $per[1', 2]$ in the table which is -1 represents the fact that r_1 lost its perfectness. Similarly 1 in the table means that r_1 was not perfect before the combination and became perfect after the combination. A value of 0 in the table means that the perfectness of r_1 did not change by the combination. To see what happens to the perfectness of r_2 when (T_1, S_1) that is in class $[1']$ and (T_2, S_2) that is in class $[2]$ are combined, we have to consider the value in the intersection of the row corresponding to class $[2]$ and the column corresponding to class $[1']$, i.e., $per[2, 1']$. In the table, the values that are not 0 are grayed out.

Table 3.4: Changes in perfectness of r_1.

per	$[1']$	$[2]$	$[4]$	$[5]$	$[6]$	$[7']$	$[8]$	$[10]$	$[11.1]$	$[11.2]$	$[12]$	$[13]$	$[14]$
$[1']$	-1	-1	-1	-1	-1	0	0	0	0	0	0	0	0
$[2]$	-1	-1	-1	-1	-1	0	0	0	0	0	0	0	0
$[4]$	-1	-1	-1	-1	-1	0	0	0	0	0	0	0	0
$[5]$	0	0	0	0	0	0	0	0	0	0	0	0	0
$[6]$	0	0	0	0	0	0	0	0	0	0	0	0	0
$[7']$	-1	-1	-1	-1	-1	0	0	0	0	0	0	0	0
$[8]$	-1	-1	-1	-1	-1	0	0	0	0	0	0	0	0
$[10]$	-1	-1	-1	-1	-1	0	0	0	0	0	0	0	0
$[11.1]$	1	1	1	1	1	0	0	0	0	0	0	0	0
$[11.2]$	0	0	0	0	0	0	0	0	0	0	0	0	0
$[12]$	1	1	1	1	1	0	0	0	0	0	0	0	0
$[13]$	1	1	1	1	1	0	0	0	0	0	0	0	0
$[14]$	0	0	0	0	0	0	0	0	0	0	0	0	0

With the introduction of the matrix per, we are now ready to modify the update step of the algorithm. For example, $vector[k, 10]$ is now computed in the update step as

$$
\begin{aligned}
vector[k,10] \coloneqq \min\{&vector[k,10] + vector[j,11.1] \\
&+ per[10,11.1]v_k + per[11.1,10]v_j, vector[k,10] \\
&+ vector[j,11.2] + per[10,11.2]v_k + per[11.2,10]v_j, \\
&vector[k,12] + vector[j,5] + per[12,5]v_k + per[5,12]v_j\}.
\end{aligned}
$$

The updated algorithm is given in Algorithm 3.

Algorithm 3 Algorithm for the MinWPNSP in trees

1: **procedure**
2: **for** $i \coloneqq 1$ to n **do**
3: initialize $vector[i]$ to $[-,-,w_i+v_i,-,-,-,-,-,-,-,0,-,-]$;
4: **end for**
5: **for** $j \coloneqq n$ down to 2 **do**
6: $k \coloneqq parent[j]$;
7: Combine the tree with root j with the tree with root k;
8: Update $vector[k]$ with the combination using "per";
9: **end for**
10: **return** $\min\{vector[1,1'], vector[1,2], vector[1,4], vector[1,5], vector[1,7'],$
11: $vector[1,8], vector[1,10], vector[1,11.1], vector[1,11.2]\}$;
12: **end procedure**

Now, we give an illustrative example. In this example, let T_0 be the tree given in Figure 3.3. The input to Algorithm 3 is the parent array which is $parent = [0,1,2,3,2]$. The value 0 in this array means that vertex 1 does not have any parent, i.e., it is the root of the tree.

Additionally, let us assume that we have the following weights.

w	0.84	0.40	0.67	-0.69	0.19
v	0.86	-0.77	-0.63	-0.05	-0.53

According to Algorithm 3, the combination operations in line 7 are done in the order given in Figure 3.4. The root of each connected component is shown with r in the

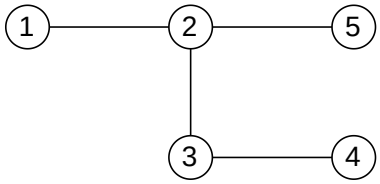


Figure 3.3: The tree used for the illustrative example.

figure.

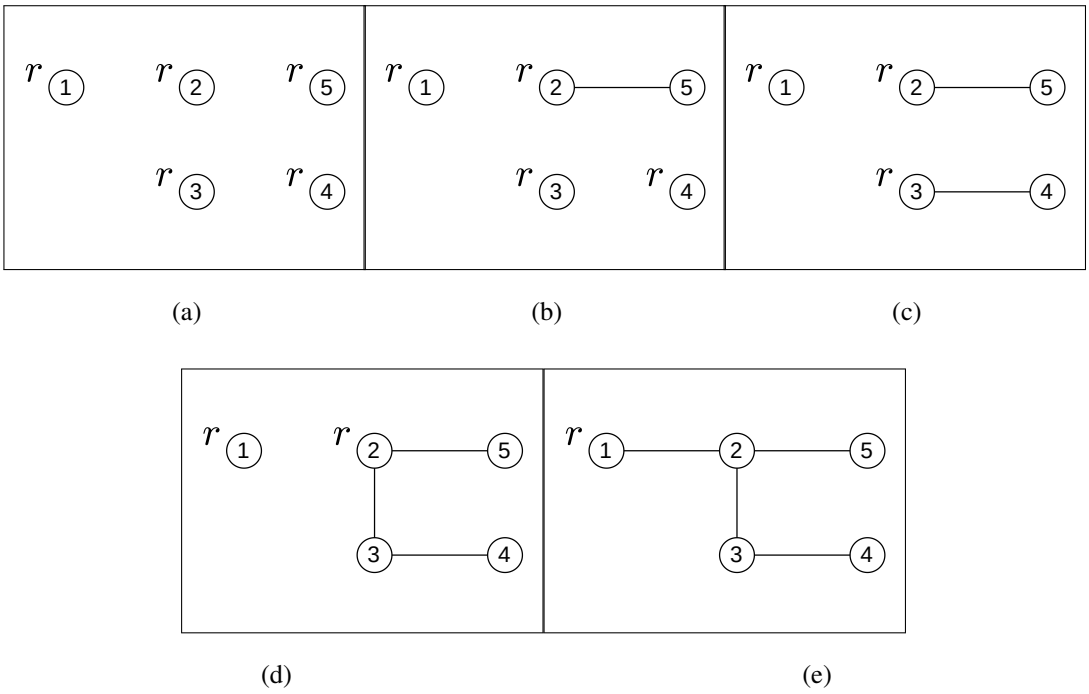


Figure 3.4: Combination operations in line 7 of Algorithm 3 for the illustrative example.

The first step of Algorithm 3 is the initialization step where the rows of $vector$ are initialized. In our example, we have the following vectors after the initialization step.

$$\begin{bmatrix} vector[1] \\ vector[2] \\ vector[3] \\ vector[4] \\ vector[5] \end{bmatrix} = \begin{bmatrix} - & - & 1.70 & - & - & - & - & - & - & - & 0 & - & - \\ - & - & -0.37 & - & - & - & - & - & - & - & 0 & - & - \\ - & - & 0.04 & - & - & - & - & - & - & - & 0 & - & - \\ - & - & -0.74 & - & - & - & - & - & - & - & 0 & - & - \\ - & - & -0.34 & - & - & - & - & - & - & - & 0 & - & - \end{bmatrix}$$

Then, according to the order "$j := 5$ down to 2", the algorithm connects the tree in which vertex 5 is the root with the tree in which its parent, vertex 2 ($parent[5] = 2$), is the root (see Figure 3.4(b)). Next, the tree in which vertex 4 is the root is connected with the tree in which its parent, vertex 3 ($parent[4] = 3$), is the root (see Figure 3.4(c)), and so on.

At the end of the execution of Algorithm 3, $vector[1]$ would be found as $vector[1] = [-, -0.44, -0.01, -, -0.03, -, -0.78, 1.35, -2.48, -, -0.35, -, -]$. The updates in $vector$ after each combination step of the algorithm are given in Table 3.5. Then, the algorithm returns min$\{vector[1, 1']$, $vector[1, 2]$, $vector[1, 4]$, $vector[1, 5]$, $vector[1, 7']$, $vector[1, 8]$, $vector[1, 10]$, $vector[1, 11.1]$, $vector[1, 11.2]\}$. In this case, the minimum value occurs for class $[11.1]$ with $vector[1, 11.1] = -2.48$. The solution with this objective function value is given in Figure 3.5.

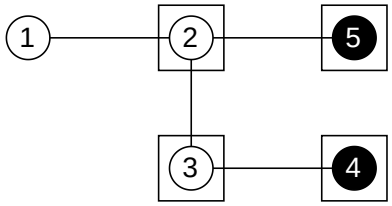


Figure 3.5: The optimal solution of the illustrative example.

Table 3.5: Updates in $vector$ after each combination step of Algorithm 3.

(a) After vertex 5 is joined with vertex 2

$vector[1]$	–	–	1.7	–	–	–	–	–	–	–	0	–	–
$vector[2]$	–	−0.9	–	–	–	–	−1.11	–	–	–	–	0	–
$vector[3]$	–	–	0.04	–	–	–	–	–	–	–	0	–	–
$vector[4]$	–	–	−0.74	–	–	–	–	–	–	–	0	–	–
$vector[5]$	–	–	−0.34	–	–	–	–	–	–	–	0	–	–

(b) After vertex 4 is joined with vertex 3

$vector[1]$	–	–	1.7	–	–	–	–	–	–	–	0	–	–
$vector[2]$	–	−0.9	–	–	–	–	−1.11	–	–	–	–	0	–
$vector[3]$	–	−0.01	–	–	–	–	−1.37	–	–	–	–	0	–
$vector[4]$	–	–	−0.74	–	–	–	–	–	–	–	0	–	–
$vector[5]$	–	–	−0.34	–	–	–	–	–	–	–	0	–	–

(c) After vertex 3 is joined with vertex 2

$vector[1]$	–	–	1.7	–	–	–	–	–	–	–	0	–	–
$vector[2]$	–	−1.64	–	0.49	–	−0.78	−2.48	–	–	−0.35	–	−1.37	–
$vector[3]$	–	−0.01	–	–	–	–	−1.37	–	–	–	–	0	–
$vector[4]$	–	–	−0.74	–	–	–	–	–	–	–	0	–	–
$vector[5]$	–	–	−0.34	–	–	–	–	–	–	–	0	–	–

(d) After vertex 2 is joined with vertex 1

$vector[1]$	–	−0.44	−0.01	–	−0.03	–	−0.78	1.35	−2.48	–	−0.35	–	–
$vector[2]$	–	−1.64	–	0.49	–	−0.78	−2.48	–	–	−0.35	–	−1.37	–
$vector[3]$	–	−0.01	–	–	–	–	−1.37	–	–	–	–	0	–
$vector[4]$	–	–	−0.74	–	–	–	–	–	–	–	0	–	–
$vector[5]$	–	–	−0.34	–	–	–	–	–	–	–	0	–	–

CHAPTER 4

AN INTEGER PROGRAMMING FORMULATION FOR THE MINWPNSP

Now, we propose an integer programming formulation for the MinWPNSP. The inputs of this solution approach are an undirected simple graph $G = (V, E)$ and weights w_j and v_j for each vertex $j \in V$. Note that the input graph G does not need to be a tree in this case. We use two sets of binary decision variables. For every vertex $j \in V$, we define

$$y_j = \begin{cases} 1, & \text{if vertex } j \text{ is in } S \\ 0, & \text{otherwise} \end{cases} \tag{4.1}$$

$$x_j = \begin{cases} 1, & \text{if vertex } j \text{ is } S\text{-perfect} \\ 0, & \text{otherwise} \end{cases} \tag{4.2}$$

With these definitions, we are now ready to give an integer programming formulation for the MinWPNSP.

$$\text{minimize} \quad \sum_{j \in V} (w_j y_j + v_j x_j) \tag{4.3}$$

$$\text{subject to} \quad \sum_{i \in \Delta(j)} x_i \geq 1 \qquad \forall j \in V \tag{4.4}$$

$$x_j \leq \sum_{i \in \Delta(j)} y_i \qquad \forall j \in V \tag{4.5}$$

$$x_j + y_\ell + y_k \leq 2 \qquad \forall j \in V, \quad \forall k, \ell \in \Delta(j), \quad k \neq \ell \tag{4.6}$$

$$x_j + \sum_{\substack{i \in \Delta(j) \\ i \neq k}} y_i \geq y_k, \qquad \forall j \in V, \quad \forall k \in \Delta(j) \tag{4.7}$$

$$x_j,\ y_j\ binary \qquad\qquad \forall j \in V \tag{4.8}$$

The objective function of this formulation minimizes the weight of S, where the set S is defined as $\{j \in V | y_j = 1\}$. The constraints enforce that S is a PN set of G.

Constraint 4.4 ensure that the set $\{j \in V | x_j = 1\}$ which is equal to $S - perfect$ dominates G. Constraints in 4.5, 4.6, and 4.7 make sure that $\{j \in V | x_j = 1\}$ is equal to $S - perfect$. For a vertex j, if $\sum_{i \in \Delta(j)} y_i = 0$, then j cannot be S-perfect. This is accomplished with Constraint 4.5 by enforcing x_j to 0. Similarly, if $\sum_{i \in \Delta(j)} y_i \geq 2$, then j cannot be S-perfect as well. In this case, x_j is forced to 0 by Constraint 4.6. Now, if $\sum_{i \in \Delta(j)} y_i = 1$, the constraints 4.5 and 4.6 become redundant and in this case Constraint 4.7 make sure that $x_j = 1$.

CHAPTER 5

CORRECTNESS OF ALGORITHM 3

When developing Algorithm 3, we have checked several times the correctness of the multiplication table that we generated. But, in any case, there is a possibility that something is overlooked. For this reason, we wanted to have another exact solution approach for the MinWPNSP for trees. Now, we compare the two solution approaches to check whether the optimal solutions returned are the same or not. Secondly, we also compare the two approaches in terms of solution time. Both solution approaches are run on a computer with Intel Core i7-4770S CPU @3.10GHz (8 CPUs) and 16.00GB RAM. We use C++ API (Visual Studio 2019, v142) for both approaches and CPLEX 12.9 to solve the integer programming formulation.

For this purpose, we consider different values for $|V| \in \{50, 100, 150, \ldots, 950, 1000\}$ by generating 1000 trees at random for each $|V|$ value. Furthermore, we generate 100 trees randomly for each $|V| \in \{2000, 3000, 4000, 5000\}$. To create a random tree, T, an algorithm similar to the Prim's algorithm to create a minimum spanning tree is applied. Prim's algorithm work by connecting two disjoint sets of the vertices with a minimum possible cost, while our algorithm work by connecting these sets with a random edge, which can be found in Algorithm 4.

The weights, both w's and v's, are selected uniformly at random from the interval $[-1, 1]$ for all instances. We observe that for each instance generated, the optimal objective function values of both solution approaches are the same. For each value of $|V|$, the average solution times of the instances are shown in Figure 5.1 for both approaches. As $|V|$ increases, the increase in average solution times of Algorithm 3 is very small and therefore it is difficult to see this change in Figure 5.1. For this reason, Figure 5.2 is provided to show the solution times of Algorithm 3 separately as well.

Algorithm 4 Algorithm to randomize a tree with $|V| = n$.

procedure $T(n)$
    initialize $T = (V, E)$ with $V = \{1, 2, \ldots, n\}$ and $E = \emptyset$.
    assing two sets as $C_1 \coloneqq \{1\}$ and $C_2 \coloneqq V - \{1\}$.
    for $i \coloneqq 1$ to $n - 1$ **do**
        Select a random item from both C_1 and C_2 as v_1 and v_2, respectively.
        $E \coloneqq E \cup \{v_1, v_2\}$.
        $C_1 \coloneqq C_1 \cup \{v_2\}$.
        $C_2 \coloneqq C_2 - \{v_2\}$.
    end for
    return $T = (V, E)$.
end procedure

From the figures, a linear trend can be seen in the average solution times of Algorithm 3. Moreover, as $|V|$ increases, the rate at which the average solution times increase gets higher for the integer programming formulation.

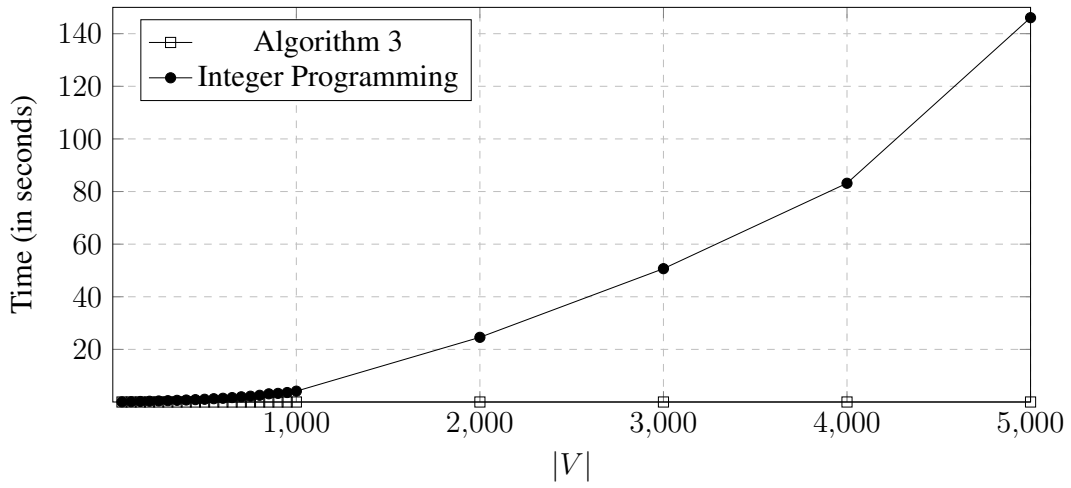


Figure 5.1: Average solution times of both solution approaches.

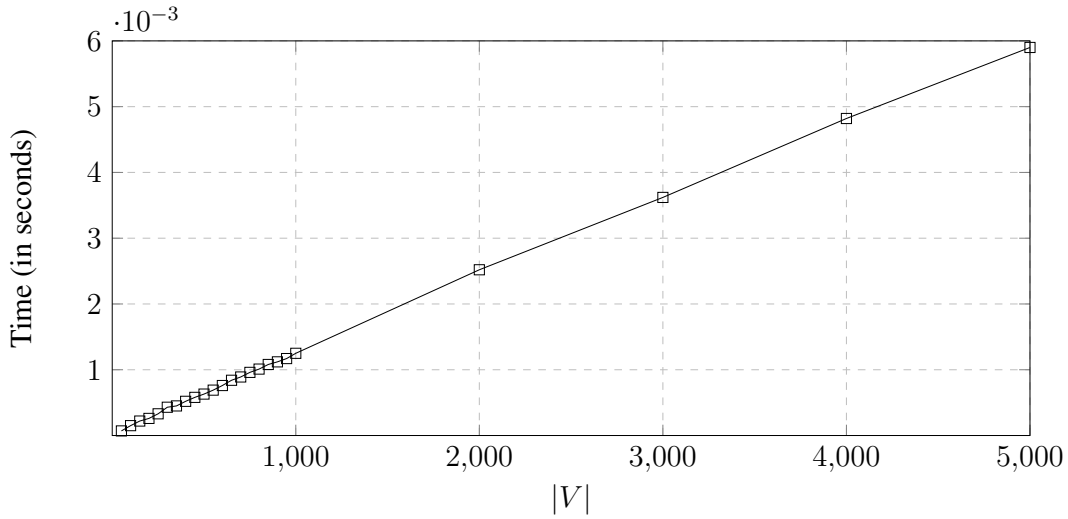


Figure 5.2: Average solution times of Algorithm 3.

CHAPTER 6

VALID INEQUALITIES

An inequality is valid for a set if it is satisfied by all points of the set. The IP formulation provided in Chapter 4 is enough to define the solution set of the MinWPNSP when the decision variables are restricted to be binary. However, when the integrality restrictions of the decision variables are relaxed, i.e. when the linear programming (LP) relaxation is considered, the optimal solution is not necessarily a feasible solution for the IP, since the LP relaxation might return a non-integral solution which would not represent a PN set. With the aim making the formulation stronger, we try to find some valid inequalities for the feasible region of the IP problem. If an inequality is valid for a polytope, we call it as a valid inequality for the problem as well. Letting P_{IP} denote the feasible region of the IP formulation, if we can find an inequality description of $conv(P_{IP})$ (which is the perfect neighborhood set polytope) by adding some valid inequalities, then the LP relaxation would return an integral solution and thus would solve the MinWPNSP.

To do this, we take the LP relaxation of the formulation in Chapter 4 as the a base model, given below.

$$
\begin{aligned}
\text{minimize} \quad & \sum_{j \in V} (w_j y_j + v_j x_j) & & \\
\text{subject to} \quad & \sum_{i \in \Delta(j)} x_i \geq 1 & & \forall j \in V \\
& x_j \leq \sum_{i \in \Delta(j)} y_i & & \forall j \in V \\
& x_j + y_\ell + y_k \leq 2 & & \forall j \in V, \quad \forall \{k, \ell\} \in \Delta(j), \quad k \neq \ell
\end{aligned}
$$

$$x_j + \sum_{i \in \Delta(j), i \neq k} y_i \geq y_k \qquad \forall j \in V, \quad \forall k \in \Delta(j)$$
$$0 \leq x_j,\ y_j \leq 1, \qquad \forall j \in V \tag{6.1}$$

We started with solving the LP relaxation for the graph given in Figure 1.1 with $w_j = 1$ and $v_j = 0$ for all $j \in V$. The solution returned is in Figure 6.1.

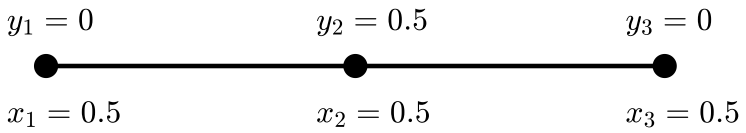


Figure 6.1: The solution of the LP relaxation of the base model for the graph given in Figure 1.1.

We, then, realized that this solution can be cut off by the following valid inequality.

$$\sum_{i \in \Delta^2(j)} y_i \geq 1, \quad \forall j \in V \tag{6.2}$$

In 6.2, $\Delta^2(j)$ represents the double neighborhood, i.e., the set of all vertices that can be reached from j within at most two edges. For the graph in Figure 6.1, we have $\Delta^2(1) = \Delta^2(2) = \Delta^2(3) = \{1, 2, 3\}$. Thus, the constraint (for any j in the graph) would be $y_1 + y_2 + y_3 \geq 1$, which cuts the current optimal solution of the LP relaxation off.

Proposition 6.1. Constraint 6.2 is a valid inequality for an arbitrary graph for the MinWPNSP.

Proof. For some vertex j of a graph, if we have $\sum_{i \in \Delta^2(j)} y_i = 0$, then we can say that $\sum_{k \in \Delta(i)} y_k = 0$ for all $i \in \Delta(j)$. Since $\sum_{k \in \Delta(i)} y_k = 0$, then we can add that $x_i = 0$ for all $i \in \Delta(j)$ and $\sum_{i \in \Delta(j)} x_i = 0$. In conclusion, this vertex is not dominated by the set of S-perfect vertices and contradicts with Constraint 4.4. □

After adding Constraint 6.2 to our model, we solved and found another non-integral solution for the following graph in Figure 6.2. Note that the weights are taken as $w_j = -1$ and $v_j = 0$ for all $j \in V$. In other words, the aim is selected to find $\Theta(G)$.

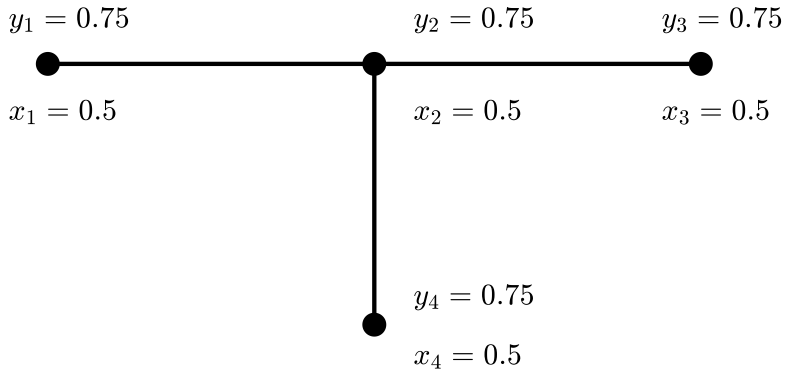


Figure 6.2: An optimal solution of the LP relaxation after the addition of Constraint 6.2.

To cut off this solution from the solution set, we came up with the following valid inequality:

$$\sum_{i \in \Delta(j)} y_i \leq |\Delta(j)| - 1, \quad \forall j \in V, \quad |\Delta(j)| \geq 2 \tag{6.3}$$

Note that for the graph given in Figure 6.2, we have $|\Delta(1)| = |\Delta(3)| = |\Delta(4)| = 2$ and $|\Delta(2)| = 4$. For the vertex 1 in the graph, Constraint 6.3 is $y_1 + y_2 \leq 1$, that cuts the current optimal solution of the LP relaxation off.

Proposition 6.2. Constraint 6.3 is a valid inequality for an arbitrary graph for the MinWPNSP.

Proof. Proof is by contradiction. Let us assume that for a vertex j of an arbitrary graph with $|\Delta(j)| \geq 2$, we have $\sum_{i \in \Delta(j)} y_i > |\Delta(j)| - 1$.

Case 1. $\sum_{i \in \Delta(j)} y_i > |\Delta(j)|$. The model restricts that $y_j \leq 1$ for all $j \in V$. If we sum this inequality for all $i \in \Delta(j)$, then we have $\sum_{i \in \Delta(j)} y_i \leq |\Delta(j)|$. Contradiction.

Case 2. $\sum_{i \in \Delta(j)} y_i = |\Delta(j)|$. This implies that all of the vertices in the closed neighborhood of vertex j is selected to be in the set S. In this case we can add $\sum_{k \in \Delta(i)} y_k \geq 2$ for all $i \in \Delta(j)$. Because it is clear that $y_j = 1$ and $y_i = 1$ for all $i \in \Delta(j)$, and $j \in \Delta(i)$ for all $i \in \Delta(j)$. Since $\sum_{k \in \Delta(i)} y_k \geq 2$, then we can add that $x_i = 0$ for all $i \in \Delta(j)$ and $\sum_{i \in \Delta(j)} x_i = 0$. This contradicts with Constraint 4.4.

□

We continued to solve the LP relaxation for some graphs after adding Constraint 6.3 to our model. The next solution with fractional optimal values is given in Figure 6.3 below.

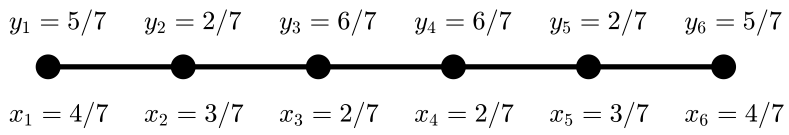


Figure 6.3: An optimal solution of the LP relaxation after the addition of Constraint 6.3.

To cut this fractional solution off, we added the following constraint into our model:

$$\sum_{\substack{i \in \Delta(j) \\ i \neq k}} y_i \leq |\Delta(j)| - 2 + x_k, \quad \forall j \in V, \quad \forall k \in \delta(j), \quad |\Delta(j)| \geq 3 \tag{6.4}$$

This constraint is written for the case where $j = 3$ and $k = 2$ since $\{2\} \in \delta(3)$ and $|\Delta(3)| = 3$. In the open form, the constraint $y_3 + y_4 \leq 1 + x_2$ cuts off the solution in Figure 6.3.

Proposition 6.3. Constraint 6.4 is a valid inequality for an arbitrary graph for the MinWPNSP.

Proof. The inequality is always valid for the case where $\sum_{i \in \Delta(j), i \neq k} y_i \leq |\Delta(j)| - 2$. If $\sum_{i \in \Delta(j), i \neq k} y_i > |\Delta(j)| - 2$, then we can say $\sum_{i \in \Delta(j), i \neq k} y_i = |\Delta(j)| - 1$ since the decision variables can be at most 1. Thus, $\sum_{t \in \Delta(i)} y_t \geq 2$ for all $i \in \Delta(j), i \neq k$ and $x_i = 0$ for all $i \in \Delta(j), i \neq k$. Constraint 4.4 implies that $\sum_{i \in \Delta(j)} x_i = x_k + \sum_{i \in \Delta(j), i \neq k} x_i \geq 1$. Since $\sum_{i \in \Delta(j), i \neq k} x_i = 0$, we have $x_k \geq 1$. In other words, vertex k must be S-perfect in all feasible solutions of this case. Overall, $\sum_{i \in \Delta(j), i \neq k} y_i \leq |\Delta(j)| - 2 + x_k$ is again valid with $|\Delta(j)| - 1 \leq |\Delta(j)| - 2 + 1$. □

Continuing in the same manner, the next fractional solution we came up with is given in Figure 6.4.

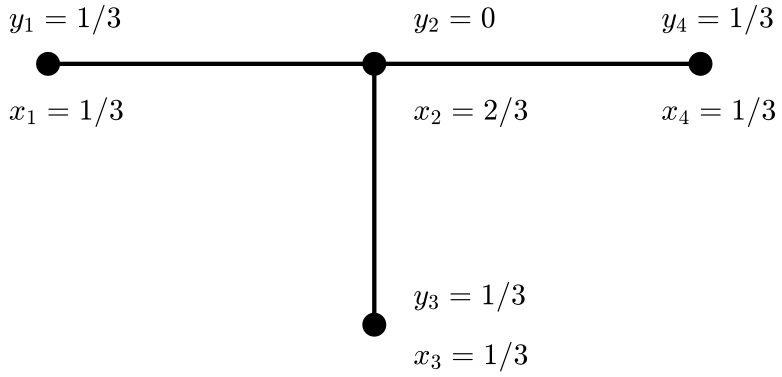


Figure 6.4: An optimal solution of the LP relaxation after the addition of Constraint 6.4.

Then, this solution is cut off by the following inequality.

$$\sum_{i \in \Delta(j)} y_i + x_j \geq 2, \quad \forall j \in Sup \tag{6.5}$$

In Figure 6.4, we have $|\Delta(1)| = |\Delta(3)| = |\Delta(4)| = 2$, thus $\{2\} \in Sup$, as Sup represents the support vertices of G. For vertex 2, Constraint 6.5 would be $y_1 + y_2 + y_3 + y_4 + x_2 \geq 2$ and cuts the current solution off.

Proposition 6.4. Constraint 6.5 is a valid inequality for an arbitrary graph for the MinWPNSP.

Proof. Let $j \in Sup$ and $k \in Pen$, as Pen represents the pendant vertices of G, that is adjacent to j. From Constraint 4.4, we have $\sum_{t \in \Delta(k)} x_t \geq 1$. Since k is a pendant vertex, Constraint 4.4 would be $x_k + x_j \geq 1$. Thus, we have either $x_k = 1$ or $x_j = 1$ (or both). In other words, we have either $\sum_{i \in \Delta(j)} y_i = 1$ or $\sum_{t \in \Delta(k)} y_t = 1$ (or both). If $\sum_{t \in \Delta(k)} y_t = 1$, then $\sum_{i \in \Delta(j)} y_i \geq 1$ since $\Delta(k) \subseteq \Delta(j)$. Thus, for any case we can say $\sum_{i \in \Delta(j)} y_i \geq 1$.

Case 1. $\sum_{i \in \Delta(j)} y_i = 1$. Then we know j is perfect, so $x_j = 1$. Therefore, $\sum_{i \in \Delta(j)} y_i + x_j = 2$.

Case 2. $\sum_{i \in \Delta(j)} y_i \geq 2$.

For both of the cases, Constraint 6.5 is satisfied. □

After adding Constraint 6.5, the following fractional solution in Figure 6.5 is found as optimal to the LP relaxation. Note that the weights of the vertices are uniformly randomized in $[-1, 1]$.

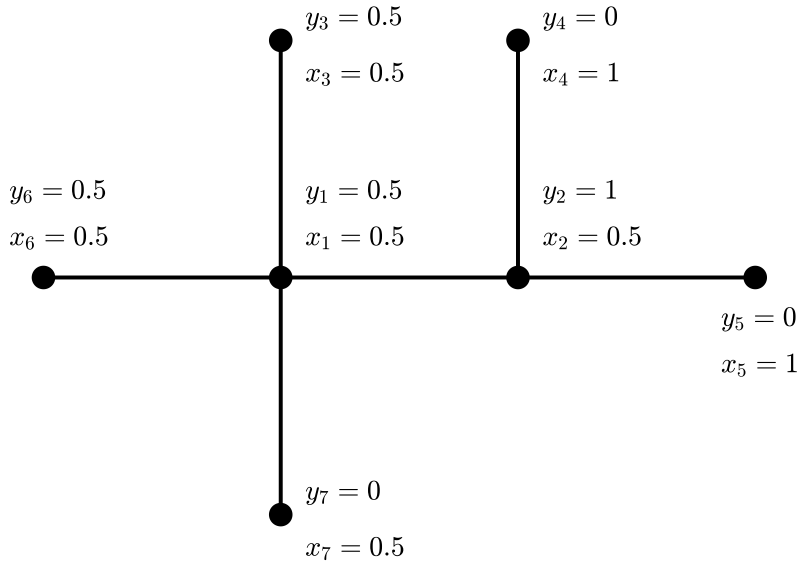


Figure 6.5: An optimal solution of the LP relaxation after the addition of Constraint 6.5.

This solution is, then, cut off by the following equation.

$$y_i + y_j = x_i, \quad \forall i \in Pen, \quad \{i, j\} \in E \tag{6.6}$$

In Equation 6.6, vertex j would be the support of vertex i. For vertex 6 of Figure 6.5, Constraint 6.6 would be $y_6 + y_1 = x_6$ that cuts the current solution off.

Proposition 6.5. Constraint 6.6 is a valid equality for an arbitrary graph for the Min-WPNSP.

Proof.

Case 1. $y_i + y_j = 2$. Then, $y_i + y_j = \sum_{k \in \Delta(i)} y_k = 2$ and $\sum_{t \in \Delta(j)} y_t \geq 2$ since $\Delta(i) \subseteq \Delta(j)$. Overall, we have $x_i = x_j = 0$.

Case 2. $y_i + y_j = 1$. Then we have $\sum_{k \in \Delta(i)} y_k = 1$, so $x_i = 1$.

Case 3. $y_i + y_j = 0$. Then we have $\sum_{k \in \Delta(i)} y_k = 0$, so $x_i = 0$.

First case cannot occur since it violates Constraint 4.4 for vertex i. For the rest of the cases, Constraint 6.6 is always satisfied. □

After adding Constraint 6.6, we came up with another fractional solution given in Figure 6.6.

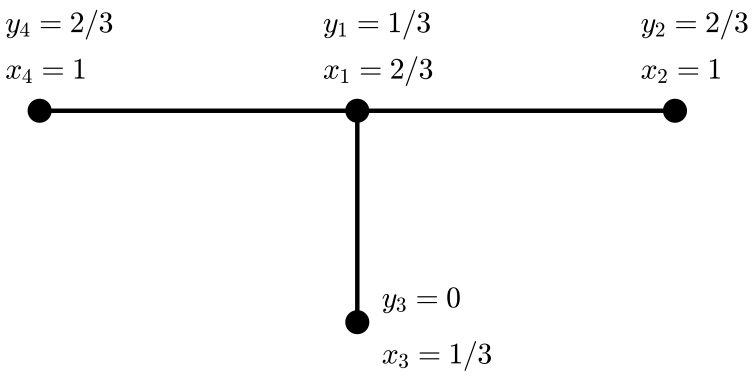


Figure 6.6: An optimal solution of the LP relaxation after the addition of Constraint 6.6.

For this graph, the following cut is used in Constraint 6.7.

$$y_j + y_k + y_\ell + x_j + x_k \leq 3, \quad \forall j \in V, \quad \forall \{k, \ell\} \in \delta(j), \quad k \neq \ell \tag{6.7}$$

Here, one of the constraints would be $y_1 + y_2 + y_4 + x_1 + x_2 \leq 3$, cutting the current solution off.

Proposition 6.6. Constraint 6.7 is a valid inequality for an arbitrary graph for the MinWPNSP.

Proof.

Case 1. $y_j + y_k + y_\ell = 3$. Then, we add that $\sum_{i \in \Delta(j)} y_i \geq 3$ and $\sum_{t \in \Delta(k)} y_t \geq 2$. Thus, $x_j = x_k = 0$. Constraint 6.7 is valid with $3 \leq 3$.

Case 2. $y_j + y_k + y_\ell = 2$. Then, $\sum_{i \in \Delta(j)} y_i \geq 2$, so $x_j = 0$. Constraint 6.7 is always valid with $2 + x_k \leq 3$.

Case 3. $y_j + y_k + y_\ell \leq 1$. For any case, we can add $x_j + x_k \leq 2$. Therefore, $y_j + y_k + y_\ell + x_j + x_k \leq 3$.

Constraint 6.7 is satisfied for all cases. □

The next fractional solution is given in Figure 6.7.

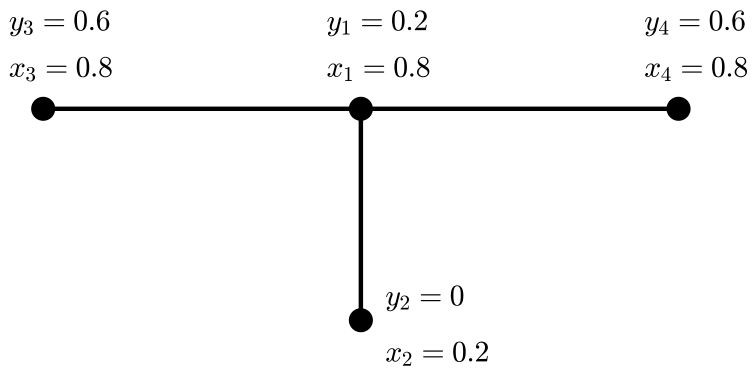


Figure 6.7: An optimal solution of the LP relaxation after the addition of Constraint 6.7.

This solution is cut off by the following inequality:

$$x_j + x_k + x_\ell \leq 2 + y_j, \quad \forall \{k, \ell\} \in Pen, \quad \forall \{k, \ell\} \in \delta(j), \quad k \neq \ell \tag{6.8}$$

For the case where $j = 1$, $k = 3$ and $\ell = 4$, we have $x_1 + x_3 + x_4 \leq 2 + y_1$. This inequality cuts the current solution off.

Proposition 6.7. Constraint 6.8 is a valid inequality for an arbitrary graph for the MinWPNSP.

Proof. If $x_j + x_k + x_\ell \leq 2$, then 6.8 is always feasible. Otherwise, we have $x_j + x_k + x_\ell = 3$. In other words, we have $\sum_{i \in \Delta(j)} y_i = \sum_{m \in \Delta(k)} y_m = \sum_{t \in \Delta(l)} y_t = 1$. Since $\{k, \ell\} \in Pen$ and $\{k, \ell\} \in \delta(j)$, we can add that $y_k + y_j = y_\ell + y_j = 1$. Thus, we have $y_k = y_\ell$.

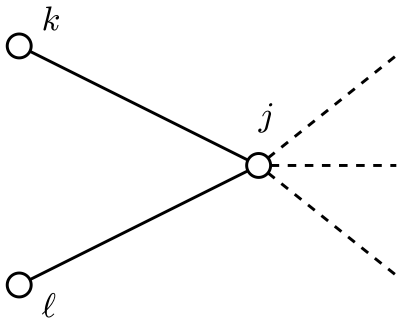


Figure 6.8: An example case for Constraint 6.8.

Case 1. $y_k = y_\ell = 0$ and $y_j = 1$. Since $\{k, \ell\} \in \delta(j)$, we can add that $\sum_{i \in \Delta(j)} y_i \geq y_j + y_k + y_\ell = 1$. Then, in order to j being S-perfect, $\sum_{i \in \Delta(j), i \neq \{j,k,\ell\}} y_i = 0$ should occur.

Case 2. $y_k = y_\ell = 1$ and $y_j = 0$. Then, we can add that $\sum_{i \in \Delta(j)} y_i \geq y_j + y_k + y_\ell = 2$. Contradiction with $\sum_{i \in \Delta(j)} y_i = 1$.

Since second case never occurs; if $x_j + x_k + x_\ell = 3$, then $y_j = 1$, which satisfies the Constraint 6.8. □

The next fractional solution after adding Constraint 6.8 into the model is given in Figure 6.9.

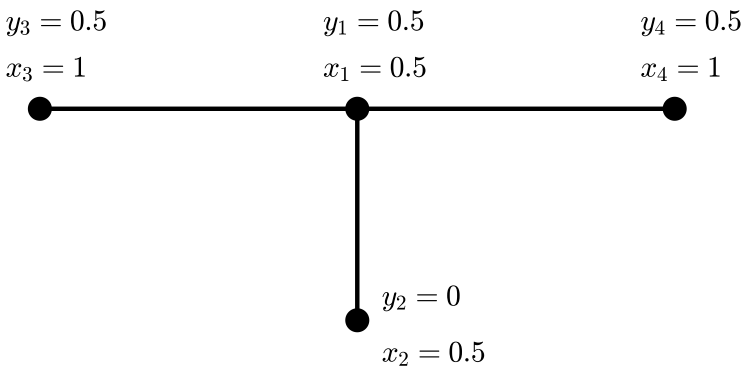


Figure 6.9: An optimal solution of the LP relaxation after the addition of Constraint 6.8.

To cut this solution off, we created the following constraint.

$$y_j + y_t + \sum_{\substack{i \in \Delta(l) \\ i \neq \{j,\ell\}}} y_i + \sum_{\substack{m \in \Delta(k) \\ m \neq \{j,k\}}} y_m + 1 \geq x_\ell + x_k, \quad \forall t \in Pen, \quad \forall \{t,k,\ell\} \in \delta(j) \tag{6.9}$$

In the graph given in Figure 6.9, for $j = 1$, $t = 2$, $k = 3$ and $\ell = 4$, Constraint 6.9 would be $y_1 + y_2 + 1 \geq x_3 + x_4$, which cuts the current solution off.

Proposition 6.8. Constraint 6.9 is a valid inequality for an arbitrary graph for the MinWPNSP.

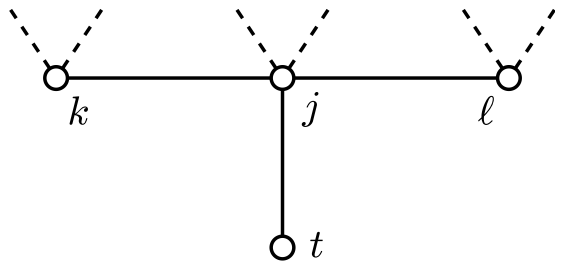


Figure 6.10: An example case for Constraint 6.9.

Proof. Constraint 6.9 is always valid if $x_\ell + x_k \leq 1$. It only cuts off the solutions with $x_\ell + x_k = 2$ and $y_j + y_t + \sum_{\substack{i \in \Delta(\ell) \\ i \neq \{j,\ell\}}} y_i + \sum_{\substack{m \in \Delta(k) \\ m \neq \{j,k\}}} y_m = 0$.

Let us assume that we have a solution with $x_\ell = x_k = 1$ and $y_j = y_t = \sum_{\substack{i \in \Delta(\ell) \\ i \neq \{j,\ell\}}} y_i = \sum_{\substack{m \in \Delta(k) \\ m \neq \{j,k\}}} y_m = 0$. Since $x_\ell = x_k = 1$, we have $\sum_{i \in \Delta(\ell)} y_i = \sum_{m \in \Delta(k)} y_m = 1$. By subtracting the sums, for example for the vertex ℓ, we have $\sum_{i \in \Delta(\ell)} y_i - \sum_{\substack{i \in \Delta(\ell) \\ i \neq \{j,\ell\}}} y_i = 1 = y_j + y_\ell$. Overall, $y_j + y_\ell = y_j + y_k = 1$. We already know that $y_j = 0$, thus $y_\ell = y_k = 1$.

Since $\sum_{p \in \Delta(j)} y_p \geq y_\ell + y_k = 2$, we have $x_j = 0$. Moreover, since $\sum_{r \in \Delta(t)} y_r = y_j + y_t = 0$, we have $x_t = 0$. By Constraint 4.4 for the vertex t, we have $\sum_{r \in \Delta(t)} x_r = x_j + x_t \geq 1$. Contradiction with $x_j + x_t = 0$. □

After adding Constraint 6.9, the LP relaxation finds the following solution optimal given in Figure 6.11.

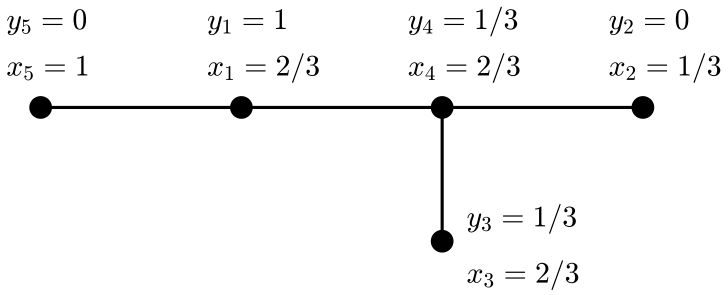


Figure 6.11: An optimal solution of the LP relaxation after the addition of Constraint 6.9.

To cut this solution off, we use Constraint 6.10.

$$x_i + x_j + y_k \leq 2, \quad \forall i \in Pen, \quad \forall \{i,k\} \in \delta(j) \tag{6.10}$$

In Figure 6.11, $\{3\} \in Pen$ and $\{1,3\} \in \delta(4)$. Thus, the constraint for this case would be $x_3 + x_4 + y_1 \leq 2$ which makes the current optimal solution infeasible.

Proposition 6.9. Constraint 6.10 is a valid inequality for an arbitrary graph for the MinWPNSP.

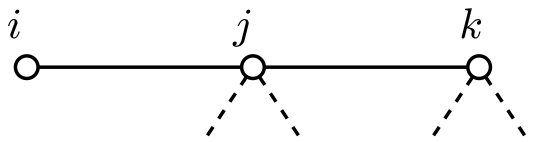


Figure 6.12: An example case for Constraint 6.10.

Proof. Let us assume that we have a solution with $x_i + x_j + y_k > 2$, i.e., $x_i = x_j = y_k = 1$. Then, we know that $\sum_{l \in \Delta(i)} y_\ell = \sum_{t \in \Delta(j)} y_t = 1$. Since $\{i\} \in Pen$ and $\{i,k\} \in \delta(j)$, we can add $\Delta(i) \subseteq \Delta(j)$ and $\{k\} \in \Delta(j) \setminus \Delta(i)$. Thus,

$\sum_{t\in\Delta(j)} y_t - \sum_{l\in\Delta(i)} y_\ell = \sum_{r\in\Delta(j)\setminus\Delta(i)} y_r = 0$. Since $\{k\} \in \Delta(j) \setminus \Delta(i)$, we can add $y_k = 0$, which contradicts with $y_k = 1$. □

The next fractional solution the model gives as output after adding Constraint 6.10 is given in Figure 6.13.

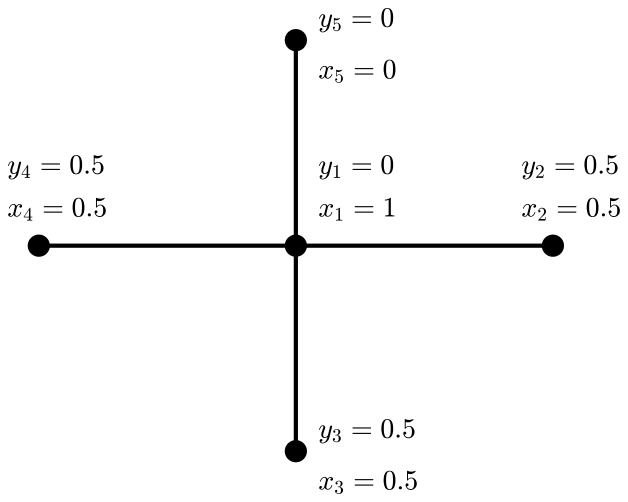


Figure 6.13: An optimal solution of the LP relaxation after the addition of Constraint 6.10.

This solution is cut off by the following constraint:

$$(|\Delta(j)| - 2)x_j + \sum_{i\in\Delta(j)} y_i \leq |\Delta(j)| - 1, \quad \forall j \in V \tag{6.11}$$

For vertex 1 in Figure 6.13, we have the constraint $3x_1 + y_1 + y_2 + y_3 + y_4 + y_5 \leq 4$, which cuts the current solution off.

This constraint, unlike the previous ones, is not created from scratch. Rather, it is a stronger version of Constraint 6.3. Additionally, this stronger version is valid for each vertex of each graph, i.e., we do not need to have a vertex with $|\Delta(j)| \geq 2$ as stated in Constraint 6.3.

Proposition 6.10. Constraint 6.11 is a valid inequality for an arbitrary graph for the MinWPNSP.

Proof.

Case 1. $x_j = 0$ and $|\Delta(j)| \geq 2$. This case has been proved to be valid in Proposition 6.2.

Case 2. $x_j = 0$ and $|\Delta(j)| = 1$. This case cannot occur with the current set of constraints because Constraint 4.4 for vertex j would be $x_j \geq 1$.

Case 3. $x_j = 1$. Then, it is clear that $\sum_{i \in \Delta(j)} y_i = 1$. Thus, the constraint is satisfied with $(|\Delta(j)| - 2) + 1 \leq |\Delta(j)| - 1$.

□

Since Constraint 6.11 is a stronger version of Constraint 6.3, the model that is to be used in the following solutions is updated by adding Constraint 6.11 to it and deleting Constraint 6.3 from it. Then, we came up with the following fractional solution.

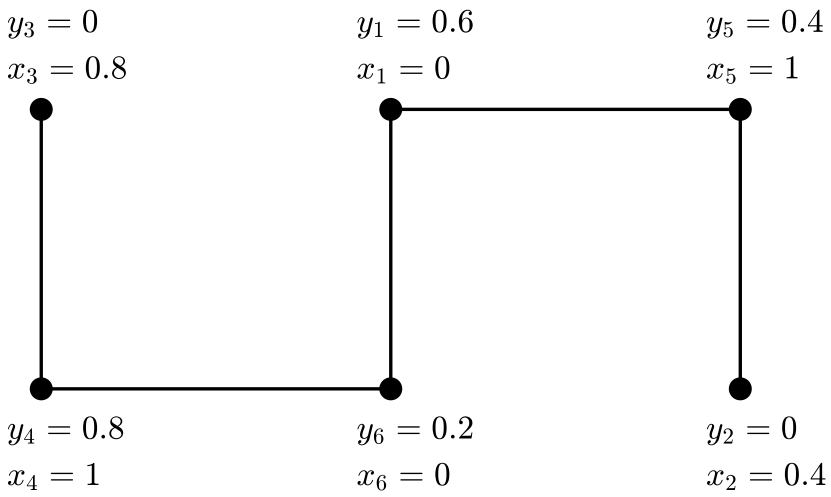


Figure 6.14: An optimal solution of the LP relaxation after the addition of Constraint 6.11.

This solution is cut off by the following inequality:

$$y_i - x_j - x_k \leq y_\ell, \quad \{\{i,j\},\{j,k\},\{k,\ell\}\} \in E, \quad |\Delta(j)| = |\Delta(k)| = 3 \qquad (6.12)$$

In the graph given in Figure 6.14, one of the constraints of 6.12 would be $y_4 - x_6 - x_1 \leq y_5$. This inequality cuts the current optimal solution off.

Proposition 6.11. Constraint 6.12 is a valid inequality for an arbitrary graph for the MinWPNSP.

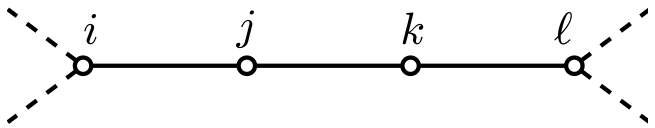


Figure 6.15: An example case for Constraint 6.12.

Proof.

Case 1. $y_i - x_j - x_k \leq 0$. Then, the constraint is always satisfied.

Case 2. $y_i - x_j - x_k > 0$. Then, the only feasibility would be $y_i - x_j - x_k = 1$ with $y_i = 1$, $x_j = 0$ and $x_k = 0$. The constraint is still satisfied if $y_\ell = 1$, so the solutions with $y_\ell = 0$ is cut off by the inequality.

Let us assume that we have a solution with $y_i = 1$, $x_j = 0$, $x_k = 0$ and $y_\ell = 0$. Constraint 4.4 for vertex j would be $x_i + x_j + x_k \geq 1$. We already know that $x_j = x_k = 0$, thus $x_i = 1$ which implies $\sum_{t \in \Delta(i)} y_t = 1$ for all feasible solutions. Since $y_i = 1$, then $\sum_{t \in \delta(i)} y_t = 0$ which implies that $y_j = 0$.

Next, we have $x_j = 0$, thus $\sum_{r \in \Delta(j)} y_r = y_i + y_j + y_k \neq 1$. When $y_i = 1$ and $y_j = 0$, the only possible case we have is $y_k = 1$. Similarly, we have $x_k = 0$, thus $\sum_{m \in \Delta(k)} y_m = y_j + y_k + y_\ell \neq 1$. Since $y_j = 0$ and $y_k = 1$, the only possible case would be $y_\ell = 1$, which contradicts with the assumption of $y_\ell = 0$. □

After adding Constraint 6.12 into our model, the solution given in Figure 6.16 is found as optimal.

This solution, then, is cut off by Constraints 6.13 and 6.14.

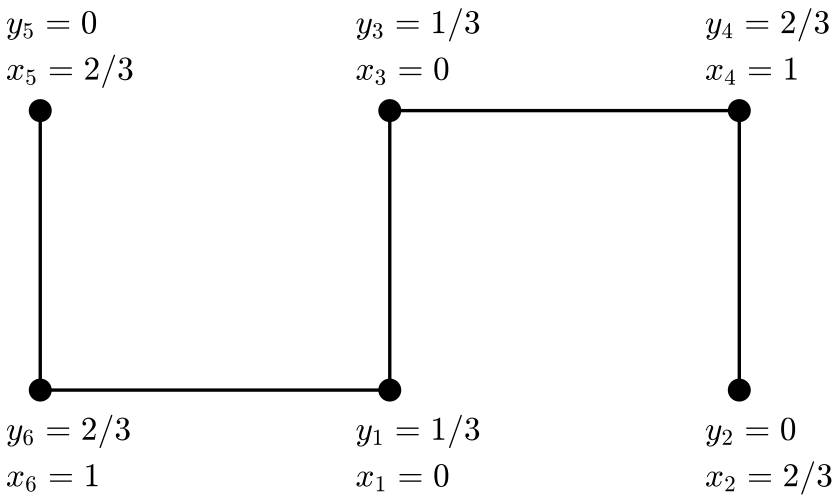


Figure 6.16: An optimal solution of the LP relaxation after the addition of Constraint 6.12.

$$y_i + y_\ell \leq 1 + x_j, \quad \{\{i,j\},\{j,k\},\{k,\ell\}\} \in E, \quad |\Delta(j)| = |\Delta(k)| = 3 \tag{6.13}$$

$$y_i + y_\ell \leq 1 + x_k, \quad \{\{i,j\},\{j,k\},\{k,\ell\}\} \in E, \quad |\Delta(j)| = |\Delta(k)| = 3 \tag{6.14}$$

Note that the case of $\{\{i,j\},\{j,k\},\{k,\ell\}\} \in E$ and $|\Delta(j)| = |\Delta(k)| = 3$ is the same of the previous case, where the example is also given in Figure 6.15. One of the constraints of 6.13 and 6.14 for the graph in Figure 6.16 would be $y_6 + y_4 \leq 1 + x_1$, which cuts the current solution off.

Proposition 6.12. Constraints 6.13 and 6.14 are valid inequalities for an arbitrary graph for the MinWPNSP.

Proof. If $y_i + y_\ell \leq 1$, then both of the constraints are always valid. If $y_i + y_\ell = 2$, then $x_j = x_k = 1$ is implied by the constraints.

Let us assume the opposite, that we have a solution with $y_i = y_\ell = 1$ and $x_j = 0$ (Note that the case where $y_i = y_\ell = 1$ and $x_k = 0$ is the same if we rename the indices by switching i with ℓ and j with k). Since $x_j = 0$, we have $y_i + y_j + y_k \neq 1$. Moreover, since $y_i = 1$, we have $y_j + y_k > 0$.

Case 1. $y_j = 1$. Constraint 4.4 for vertex j implies that $x_i + x_j + x_k \geq 1$. Since we have $y_i = y_j = y_\ell = 1$, all of the vertices of i, j and k has at least two vertices from set S (as y's) in their closed neighborhoods, making them as not S-perfect vertices, i.e., $x_i + x_j + x_k = 0$. Contradiction.

Case 2. $y_k = 1$. This case is very similar to the previous case. Since $y_i = y_k = y_\ell = 1$, we conclude that $x_j + x_k + x_\ell = 0$. Contradiction with Constraint 4.4 for vertex k.

Since none of the cases are valid, any solution with $y_i = y_\ell = 1$ and $x_j = 0$ (or $x_k = 0$) is infeasible for the base formulation. □

After adding constraints 6.13 and 6.14 into our model, we came up with the following fractional solution:

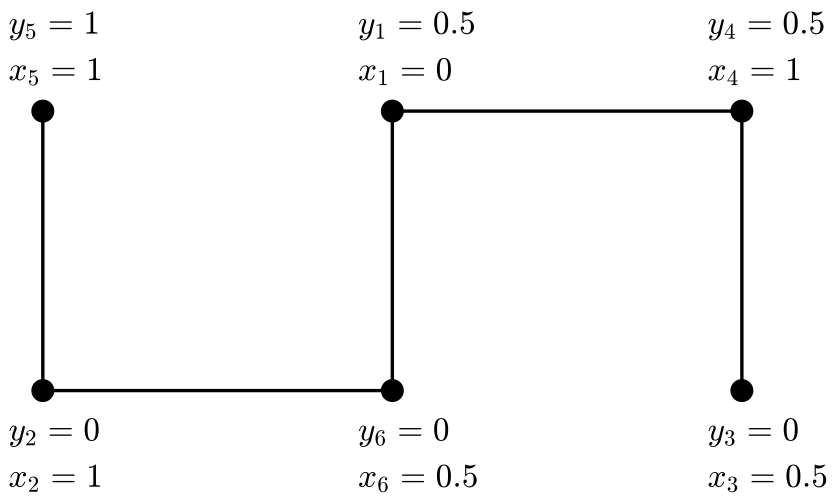


Figure 6.17: An optimal solution of the LP relaxation after the addition of Constraint 6.13 and 6.14.

To cut the fractional solution in Figure 6.17 off, we used Constraint 6.15:

$$y_j + x_k \leq 1 + y_i + x_j, \quad \{\{i,j\},\{j,k\}\} \in E, \quad |\Delta(j)| = 3 \tag{6.15}$$

One of the constraints of inequality 6.15 would be $y_1 + x_4 \leq 1 + y_6 + x_1$ for the graph in Figure 6.17, which cuts off the current solution of it.

Proposition 6.13. Constraint 6.15 is a valid inequality for an arbitrary graph for the MinWPNSP.

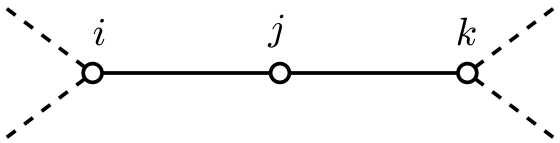


Figure 6.18: An example case for Constraint 6.15.

Proof. Constraint 6.15 would be infeasible if and only if $y_j = x_k = 1$ and $y_i = x_j = 0$. Let us assume we have such a solution. In this case, it is clear that $y_k = 1$, because otherwise x_j would have been 1. Then, since $y_j = y_k = 1$, we can say that $x_k = 0$. Contradiction with the initial assumption as $x_k = 1$. □

Similarly, we added Constraint 6.15 into our model and tried to find other fractional solutions. One of them is given below in Figure 6.19:

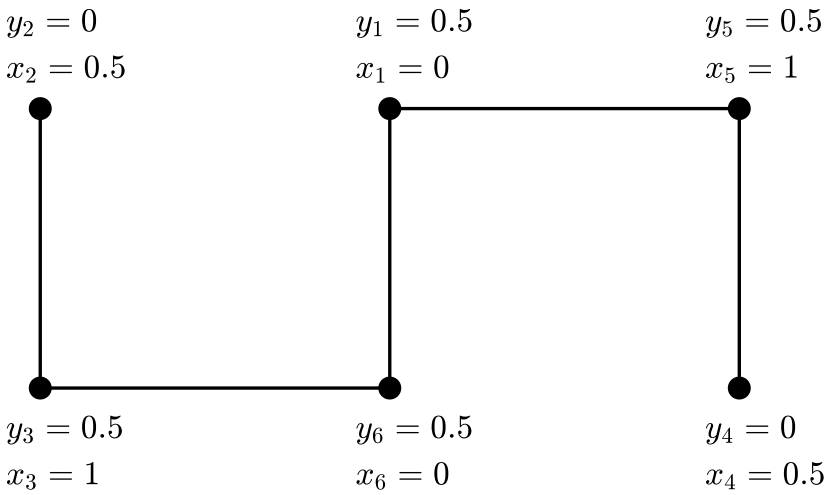


Figure 6.19: An optimal solution of the LP relaxation after the addition of Constraint 6.15.

To cut this solution off, the following constraint is used:

$$x_j \leq y_i + y_\ell + x_k, \quad \{\{i,j\},\{j,k\},\{k,\ell\}\} \in E, \quad |\Delta(j)| = |\Delta(k)| = 3 \qquad (6.16)$$

One of Constraint 6.16 for the graph in Figure 6.19 would be $x_3 \leq y_2 + y_1 + x_6$ which cuts off the current solution.

Proposition 6.14. Constraint 6.16 is a valid inequality for an arbitrary graph for the MinWPNSP.

Proof. Constraint 6.16 is infeasible if and only if $x_j = 1$ and $y_i = y_\ell = x_k = 0$. Let us assume we have such a solution. Since $x_j = 1$, we know that $y_i + y_j + y_k = 1$. Moreover, since $y_i = 0$, we add that $y_j + y_k = 1$. If we sum both sides of this inequality by y_ℓ which is 0, we will have that $y_j + y_k + y_\ell = 1$. This equation implies that $x_k = 1$ since $|\Delta(k)| = \{j,k,\ell\}$. Contradiction with our initial assumption of $x_k = 0$. □

With the addition of Constraint 6.16, one of the fractional solutions the model returned is given in Figure 6.20:

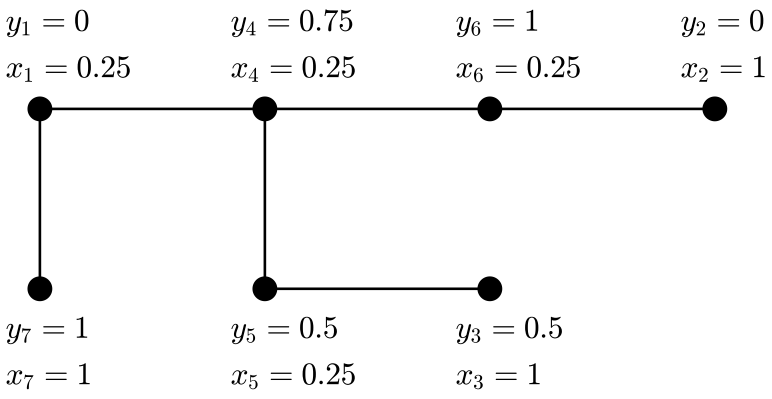


Figure 6.20: An optimal solution of the LP relaxation after the addition of Constraint 6.16.

To cut the solution given in Figure 6.20 off, we used Constraint 6.17.

$$(|\Delta(j)| - 3)x_j + \sum_{\substack{i \in \Delta(j) \\ i \neq k}} y_i \leq |\Delta(j)| - 2 + x_k, \quad \forall j \in V, \quad \forall k \in \delta(j), \quad |\Delta(j)| \geq 3 \tag{6.17}$$

This constraint is a stronger version of Constraint 6.4. The left hand side is added with $(|\Delta(j)| - 3)x_j$. In Figure 6.20, we have $|\Delta(4)| = 4$. Thus, for $j = 4$ and $k = 1$, Constraint 6.17 would be $x_4 + y_4 + y_5 + y_6 \leq 2 + x_1$. This makes the solution in Figure 6.20 infeasible.

Proposition 6.15. Constraint 6.17 is a valid inequality for an arbitrary graph for the MinWPNSP.

Proof.

Case 1. $x_j = 0$. This case has been proved to be valid in Proposition 6.3.

Case 2. $x_j = 1$. Then, we know that $\sum_{i \in \Delta(j)} y_i = 1$ and $\sum_{i \in \Delta(j), i \neq k} y_i \leq 1$. If we sum both sides with $(|\Delta(j)| - 3)x_j$, we have that $(|\Delta(j)| - 3)x_j + \sum_{i \in \Delta(j), i \neq k} y_i \leq (|\Delta(j)| - 2)$. In this case, Constraint 6.17 is always valid.

□

Similar to the Constraints 6.3 and 6.11, at this point, the model is updated by adding Constraint 6.17 to it and by deleting Constraint 6.4 from it for further solutions. Then, the solution in Figure 6.21 is found as optimal.

To cut this solution off, the following constraint is used:

$$(|\Delta(j)| - 2)x_j + \sum_{i \in \Delta(j)} y_i = |\Delta(j)| - 1, \quad \forall j \in Sup \tag{6.18}$$

where all the other vertices in the set of $V - \{j\}$ should be pendant vertices and adjacent to j. In other words, the graph must be a star graph. Therefore, this is the first constraint used in this chapter which is not valid in general for an arbitrary graph.

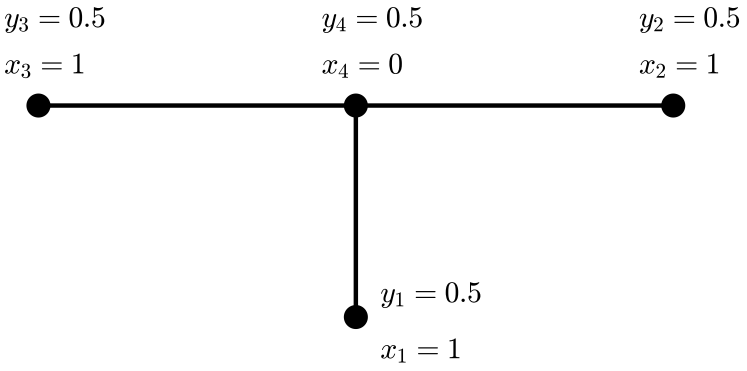


Figure 6.21: An optimal solution of the LP relaxation after the addition of Constraint 6.17.

Since the graph in Figure 6.21 is a star graph, Constraint 6.18 would be $2x_4 + y_1 + y_2 + y_3 + y_4 = 3$. The current solution, then, would be infeasible with $2 \neq 3$ by adding Constraint 6.18.

Proposition 6.16. Constraint 6.18 is a valid equality for a star graph for the MinWP-NSP.

Proof. Since j is the support of the star graph, we have $|\Delta(j)| = n$ and $\Delta(j) = V$.

Case 1. $x_j = 1$. Then, it is known that $\sum_{i \in V} y_i = 1$. Thus the sum would be $(n-2) + 1 = (n-1)$.

Case 2. $x_j = 0$. Since for a vertex $i \in V - \{j\}$, we have $x_i + x_j \geq 1$ by Constraint 4.4, we add that $x_i = 1$ for all $i \in V - \{j\}$. In other words, $y_i + y_j = 1$. Here, y_j cannot be 1 since if $y_j = 1$, then $y_i = 0$ for all $i \in V - \{j\}$, which implies $\sum_{k \in V} y_k = 1$ and $x_j = 1$. This contradicts with our assumption of $x_j = 0$. Thus, we add that $y_j = 0$ and $y_i = 1$ for all $i \in V - \{j\}$. In this case, the constraint would be satisfied with $(n-2)0 + (n-1) = (n-1)$.

The constraint is valid for both cases. □

After adding Constraint 6.18 into our model, the solution given in Figure 6.22 is found as optimal by the model.

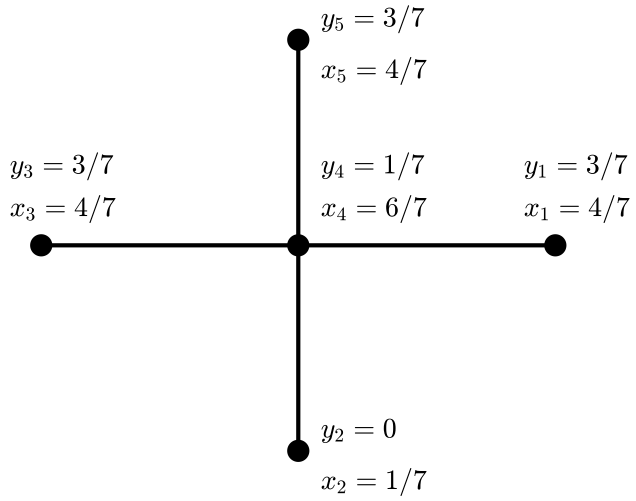


Figure 6.22: An optimal solution of the LP relaxation after the addition of Constraint 6.18.

Then, we use Constraint 6.19 to cut off the solution given in Figure 6.22.

$$x_j + y_i \geq 1, \quad |\Delta(j)| = n, \quad |\Delta(i)| = 2 \tag{6.19}$$

This constraint would be written for $i = 2$ and $j = 4$ for the graph in Figure 6.22 since we have $|\Delta(2)| = 2$ and $|\Delta(4)| = 5$. Then, $x_4 + y_2 \geq 1$ would cut the current optimal solution off.

Proposition 6.17. Constraint 6.19 is a valid equality for an arbitrary graph for the MinWPNSP.

Proof. The constraint would cut off the solutions with $x_j + y_i < 1$. In other words, $x_j = y_i = 0$. Let us assume that we have a solution with $x_j = y_i = 0$. Since $|\Delta(j)| = n$ and $|\Delta(i)| = 2$, we know that i is a pendant vertex and $\{i, j\} \in E$. Moreover, by Constraint 4.4 for vertex i, we have the condition that $x_i + x_j \geq 1$. Since $x_j = 0$, we have $x_i = 1$ which implies $y_i + y_j = 1$. Since also $y_i = 0$, we know that $y_j = 1$. Additionally, $x_j = 0$ and $y_j = 1$ imply that $\sum_{k \in V} y_k \neq 1$ and $\sum_{k \in V} y_k \geq 1$, respectively. Overall, we have that $\sum_{k \in V} y_k \geq 2$.

Case 1. $n > 2$. Then, there exists at least one vertex, say t, in the set of $V - \{i, j\}$ which has $y_t = 1$. Since $|\Delta(j)| = n$, we can add that $|\Delta(t)| \subseteq |\Delta(j)|$. Thus, any

vertex in $|\Delta(t)|$ would also be adjacent to vertex j. Since we have $y_j = y_t = 1$, we can say that $\sum_{m \in \Delta(\ell)} y_m \geq 2$ for all $\ell \in \Delta(t)$, implying $x_\ell = 0$ for all $\ell \in \Delta(t)$. This would be infeasible since Constraint 4.4 for vertex t implies that $\sum_{\ell \in \Delta(t)} x_\ell \geq 1$.

Case 2. $n \leq 2$. We assumed that we have two distinct vertices i and j, thus $n \geq 2$, implying $n = 2$. Thus, $\sum_{k \in V} y_k \geq 2$ converts into $y_i + y_j \geq 2$. This is not possible since we initially assumed that $y_i = 0$.

□

Addition of Constraint 6.19 into our model gives us another fractional solution as follows:

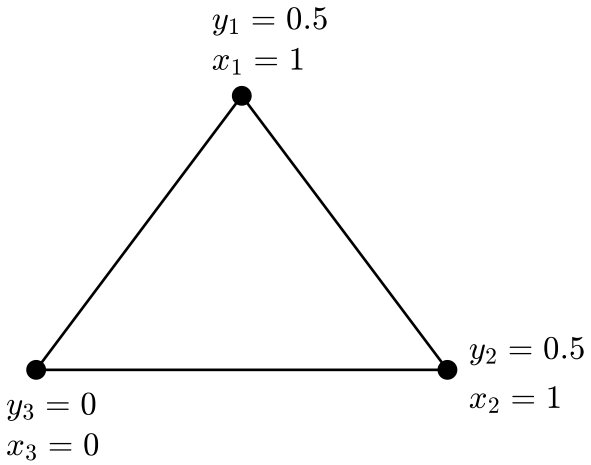


Figure 6.23: An optimal solution of the LP relaxation after the addition of Constraint 6.19.

The solution in Figure 6.23 is, then, cut off by the following constraint, where $C \subseteq V$ is a maximal clique:

$$x_j = \sum_{i \in C} y_i, \quad \Delta(j) = C \tag{6.20}$$

For the graph in Figure 6.20, we have one maximal clique as $C = \{1, 2, 3\}$ and $\Delta(1) = \Delta(2) = \Delta(3) = C$. Thus, one of the constraints of the new equation would imply that $x_3 = y_1 + y_2 + y_3$, which cuts the current fractional solution off.

Proposition 6.18. Constraint 6.20 is a valid equality for an arbitrary graph for the MinWPNSP.

Proof.

Case 1. $\sum_{i \in C} y_i = 0$. Then, since $\Delta(j) = C$, we have $\sum_{i \in \Delta(j)} y_i = 0$, implies $x_j = 0 = \sum_{i \in C} y_i$.

Case 2. $\sum_{i \in C} y_i = 1$. Then, vertex j would be perfect by the definition since $\sum_{i \in C} y_i = \sum_{i \in \Delta(j)} y_i = 1$, thus $x_j = 1 = \sum_{i \in C} y_i$.

Case 3. $\sum_{i \in C} y_i \geq 2$. Then, $x_j = 0$ since $\sum_{i \in C} y_i = \sum_{i \in \Delta(j)} y_i \geq 2$. By the Constraint 4.4 for vertex j, we add that there exists at least one vertex $k \in \Delta(j)$ which is perfect, $x_k = 1$ for each feasible solution of the problem. We also know that $k \in \Delta(j) = C$ and $C \subseteq \Delta(k)$ since C is a clique and k should be adjacent to each vertex of the clique that is consisted of. Thus, we have $\sum_{i \in \Delta(k)} y_i \geq \sum_{i \in C} y_i \geq 2$, implying $x_k = 0$ since $\sum_{i \in \Delta(k)} y_i \neq 1$. Contradiction with $x_k = 1$.

Constraint 6.20 is valid for the first two cases and last case is not possible to occur for any feasible solution of the problem. □

Constraint 6.20 implies Constraint 6.6 because 6.6 is the same of 6.20 when $|C| = 2$. Thus, we updated our model by adding Constraint 6.20 and removing Constraint 6.6 for the further solutions. After this update, we came up with the fractional solution in Figure 6.24.

To cut this solution off, we created the following constraint:

$$x_i = x_j, \quad \Delta(i) = \Delta(j) \tag{6.21}$$

For the graph in Figure 6.24, Constraint 6.21 implies that $x_2 = x_3$ since $\Delta(2) = \Delta(3) = \{1, 2, 3, 4\}$ which cuts off the current fractional solution.

Proposition 6.19. Constraint 6.21 is a valid equality for an arbitrary graph for the MinWPNSP.

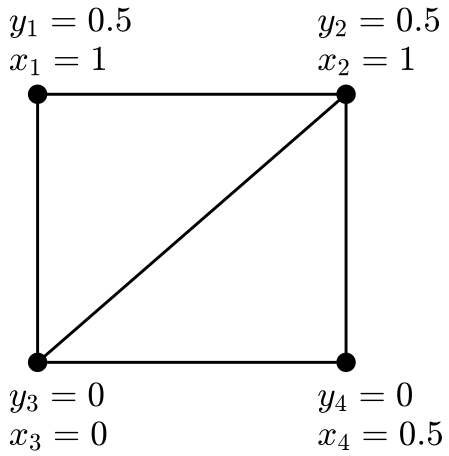


Figure 6.24: An optimal solution of the LP relaxation after the addition of Constraint 6.20.

Proof. If $x_i = 1$, then $\sum_{k \in \Delta(i)} y_k = 1$, thus $\sum_{m \in \Delta(j)} y_m = 1$ since $\Delta(i) = \Delta(j)$, which implies $x_j = 1 = x_i$. Else, $x_i = 0$ implies $\sum_{k \in \Delta(i)} y_k \neq 1$, thus $\sum_{m \in \Delta(j)} y_m \neq 1$ since $\Delta(i) = \Delta(j)$, which implies $x_j = 0 = x_i$. □

Continuing in the same manner, we had the following fractional solution after adding Constraint 6.21 into our model:

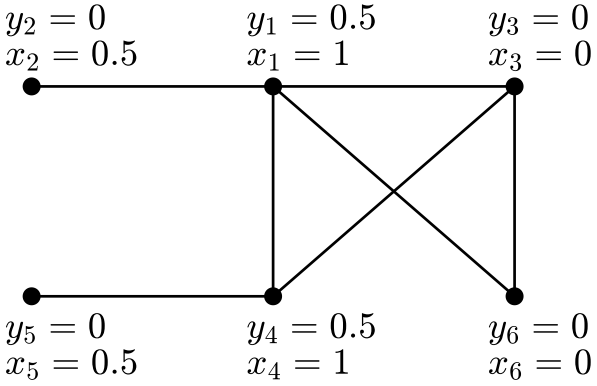


Figure 6.25: An optimal solution of the LP relaxation after the addition of Constraint 6.21.

For the solution in Figure 6.25, we came up with the constraint $y_1 + x_1 \leq 1 + x_3$ which cuts off the current optimal. In the general form, the constraint is

$$y_k + x_j \leq 1 + x_i, \quad k \in \Delta(i) \subseteq \Delta(j) \tag{6.22}$$

Proposition 6.20. Constraint 6.22 is a valid inequality for an arbitrary graph for the MinWPNSP.

Proof. The constraint is always feasible if $y_k + x_j \leq 1$. If $y_k = x_j = 1$, it implies that $x_i = 1$. Thus, it only cuts the subset of solutions in which $y_k = x_j = 1$ and $x_i = 0$.

Let us assume that we have a solution in which $y_k = x_j = 1$ and $x_i = 0$. $x_j = 1$ implies that $\sum_{\ell \in \Delta(j)} y_\ell = 1$ and $\sum_{m \in \Delta(i)} y_m \leq 1$ since $\Delta(i) \subseteq \Delta(j)$. Additionally, we have $y_k = 1$ and $k \in \Delta(i)$, which implies $\sum_{m \in \Delta(i)} y_m \geq 1$. By the combination of the results, we have $\sum_{m \in \Delta(i)} y_m = 1$, implying $x_i = 1$. This contradicts with our initial assumption of $x_i = 0$. □

With the addition of Constraint 6.22 in our model, we came up with the fractional optimal solution in Figure 6.26.

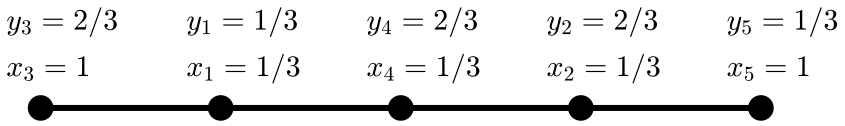


Figure 6.26: An optimal solution of the LP relaxation after the addition of Constraint 6.22.

To cut this solution off, the following constraint is used:

$$y_i + y_j + y_k + y_\ell \leq 2, \quad \{\{i,j\},\{j,k\},\{k,\ell\}\} \in E, \quad |\Delta(j)| = |\Delta(k)| = 3 \tag{6.23}$$

In Figure 6.26, one of the constraints of this form would be $y_3 + y_1 + y_4 + y_2 \leq 2$, which cuts the current optimal solution off. (Note that the demonstration of Constraint 6.23 is given in Figure 6.15)

Proposition 6.21. Constraint 6.23 is a valid inequality for an arbitrary graph for the MinWPNSP.

Proof. Let us assume that we have a solution of $y_i + y_j + y_k + y_\ell > 2, \quad \{\{i,j\}, \{j,k\}, \{k,\ell\}\} \in E, \quad |\Delta(j)| = |\Delta(k)| = 3.$

Case 1. $y_i = 1$, $y_j = 1$, $y_k = 1$, and $y_\ell = 0$. This contradicts with Constraint 6.11 for j which implies $x_j + \sum_{m \in \Delta(j)} y_m = x_j + y_i + y_j + y_k \leq |\Delta(j)| - 1 = 2$.

Case 2. $y_i = 1$, $y_j = 1$, $y_k = 0$, and $y_\ell = 1$. This implies that $\sum_{m \in \Delta(i)} y_m \geq 2$, $\sum_{m \in \Delta(j)} y_m = 2$, and $\sum_{m \in \Delta(k)} y_m = 2$, implying $x_i = x_j = x_k = 0$. This contradicts with Constraint 4.4 for j which implies that $\sum_{m \in \Delta(j)} x_m = x_i + x_j + x_k \geq 1$.

Case 3. $y_i = 1$, $y_j = 0$, $y_k = 1$, and $y_\ell = 1$ (Symmetric case with Case 2). This implies that $\sum_{m \in \Delta(j)} y_m = 2$, $\sum_{m \in \Delta(k)} y_m = 2$, and $\sum_{m \in \Delta(\ell)} y_m \geq 2$, implying $x_j = x_k = x_\ell = 0$. This contradicts with Constraint 4.4 for k which implies that $\sum_{m \in \Delta(k)} x_m = x_j + x_k + x_\ell \geq 1$.

Case 4. $y_i = 0$, $y_j = 1$, $y_k = 1$, and $y_\ell = 1$ (Symmetric case with Case 1). This contradicts with Constraint 6.11 for k which implies $x_k + \sum_{m \in \Delta(k)} y_m = x_k + y_j + y_k + y_\ell \leq |\Delta(k)| - 1 = 2$.

Case 5. $y_i = 1$, $y_j = 1$, $y_k = 1$, and $y_\ell = 1$. This contradicts with Constraint 6.11 for j which implies $x_j + \sum_{m \in \Delta(j)} y_m = x_j + y_i + y_j + y_k \leq |\Delta(j)| - 1 = 2$.

All of the cases contradict with the current set of constraints. □

With the addition of Constraint 6.23 into our model, the next fraction solution the updated model returned is given in Figure 6.27

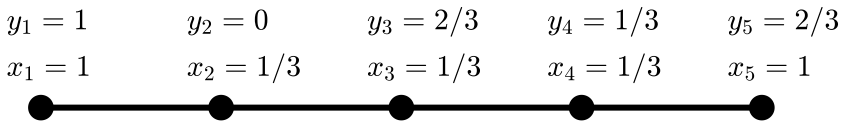


Figure 6.27: An optimal solution of the LP relaxation after the addition of Constraint 6.23.

A valid constraint as $y_1 + y_4 \leq x_2 + x_3$ cuts off the current optimal solution. In the general form, the constraint is

$$y_i + y_\ell \leq x_j + x_k, \quad \{\{i,j\},\{j,k\},\{k,\ell\}\} \in E, \quad |\Delta(j)| = |\Delta(k)| = 3 \tag{6.24}$$

Here, for $i = 1$, $j = 2$, $k = 3$ and $\ell = 4$, the constraint is generated since $|\Delta(2)| = |\Delta(3)| = 3$ and these vertices are adjacent to each other for each consecutive pair. Thus, the current optimal solution is successfully cut.

Proposition 6.22. Constraint 6.24 is a valid inequality for an arbitrary graph for the MinWPNSP.

Proof.

Case 1. $y_i + y_\ell = 0$. Then, $x_j + x_k \geq 0$ is always satisfied and Constraint 6.24 is redundant.

Case 2. $y_i + y_\ell = 1$. Since the case of $\{\{i,j\},\{j,k\},\{k,\ell\}\} \in E$ and $|\Delta(j)| = |\Delta(k)| = 3$ is symmetric (see Figure 6.15), we assume that $y_i = 1$ and $y_\ell = 0$ without loss of generality. In such a case, Constraint 6.24 implies that $x_j + x_k \geq 1$, so it cuts off the solutions with $x_j + x_k = 0$.

Let us assume that we have a feasible solution with $y_i = 1$, $x_j = 0$, $x_k = 0$ and $y_\ell = 0$. $x_j = 0$ implies that $\sum_{m \in \Delta(j)} y_m = y_i + y_j + y_k \neq 1$. Since $y_i = 1$, we have $y_j + y_k \neq 0$ by combining these two results. It has already been proved that $y_j + y_k = 2$ is infeasible for this case (see Case 1 of Proposition 6.21). Thus, we only have one possibility as $y_j + y_k = 1$. Then, one can observe that the sum of y values in the closed neighborhood of vertex k is 1, i.e., $\sum_{t \in \Delta(k)} y_t = y_j + y_k + y_\ell = 1$, implying $x_k = 1$. This contradicts with out initial assumption as $x_k = 0$.

Case 3. $y_i + y_\ell = 2$. The model implies that $y_i + y_j + y_k + y_\ell \leq 2$ by Constraint 6.23, thus we have that $y_j = y_k = 0$. Since $\sum_{m \in \Delta(j)} y_m = y_i + y_j + y_k = 1 + 0 + 0 = 1$, we have $x_j = 1$. Similarly, since $\sum_{t \in \Delta(k)} y_t = y_j + y_k + y_\ell = 0 + 0 + 1 = 1$, we also have $x_k = 1$. So, $y_i + y_\ell \leq x_j + x_k$ is satisfied with $2 \leq 2$.

Overall, Constraint 6.24 is always valid for an arbitrary graph for the MinWPNSP. □

Constraint 6.24 is stronger than Constraint 6.12, 6.13 and 6.14, these constraints would always be redundant by the addition of 6.24. Constraint 6.12 implies that $y_i - x_j - x_k \leq y_\ell$ for a case as in Figure 6.15. In other words, it implies $y_i + y_\ell \leq 2y_\ell + x_j + x_k$. It is clear that this is always satisfied by the current model where all of the feasible solutions satisfy Constraint 6.24, which implies $y_i + y_\ell \leq x_j + x_k$ for the same case in Figure 6.15 and we have $x_j + x_k \leq 2y_\ell + x_j + x_k$. Similarly, Constraint 6.24 also implies 6.13 and 6.14 since $y_i + y_\ell \leq x_j + x_k \leq 1 + x_k$ and $y_i + y_\ell \leq x_j + x_k \leq 1 + x_j$ are valid when decision variables are bounded to be at most 1.

At this point, we stopped adding constraints into our model. After the elimination of the implied constraints, the finalized model for the MinWPNSP can be found as follows.

$$\text{minimize} \quad \sum_{j \in V} (w_j y_j + v_j x_j)$$

subject to

$$\sum_{i \in \Delta(j)} x_i \geq 1 \qquad \forall j \in V$$

$$x_j \leq \sum_{i \in \Delta(j)} y_i \qquad \forall j \in V$$

$$x_j + y_\ell + y_k \leq 2 \qquad \forall j \in V, \quad \forall \{k, \ell\} \in \Delta(j), \quad k \neq \ell$$

$$x_j + \sum_{i \in \Delta(j), i \neq k} y_i \geq y_k \qquad \forall j \in V, \quad \forall k \in \Delta(j)$$

$$x_j,\ y_j \ binary \qquad \forall j \in V$$

$$\sum_{i \in \Delta^2(j)} y_i \geq 1 \qquad \forall j \in V$$

$$\sum_{i \in \Delta(j)} y_i + x_j \geq 2 \qquad \forall j \in Sup$$

$$y_j + y_k + y_\ell + x_j + x_k \leq 3 \qquad \forall j \in V, \quad \forall \{k, \ell\} \in \delta(j), \quad k \neq \ell$$

$$x_j + x_k + x_\ell \leq 2 + y_j \qquad \forall \{k, \ell\} \in Pen, \quad \forall \{k, \ell\} \in \delta(j), \quad k \neq \ell$$

$$y_j + y_t + \sum_{\substack{i \in \Delta(l) \\ i \neq \{j, \ell\}}} y_i + \sum_{\substack{m \in \Delta(k) \\ m \neq \{j, k\}}} y_m + 1 \geq x_\ell + x_k, \quad \forall t \in Pen, \quad \forall \{t, k, \ell\} \in \delta(j)$$

$$x_i + x_j + y_k \leq 2 \qquad \forall i \in Pen, \quad \forall \{i, k\} \in \delta(j)$$

$$
\begin{aligned}
&(|\Delta(j)|-2)x_j + \sum_{i\in\Delta(j)} y_i \le |\Delta(j)|-1, \quad \forall j \in V \\
&y_j + x_k \le 1 + y_i + x_j && \{\{i,j\},\{j,k\}\} \in E, \quad |\Delta(j)| = 3 \\
&x_j \le y_i + y_\ell + x_k && \{\{i,j\},\{j,k\},\{k,\ell\}\} \in E, \quad |\Delta(j)| = |\Delta(k)| = 3 \\
&(|\Delta(j)|-3)x_j + \sum_{\substack{i\in\Delta(j)\\ i\neq k}} y_i \le |\Delta(j)|-2+x_k, \quad \forall j \in V, \quad \forall k \in \delta(j), \quad |\Delta(j)| \ge 3 \\
&(|\Delta(j)|-2)x_j + \sum_{i\in\Delta(j)} y_i = |\Delta(j)|-1, \quad \forall j \in Sup \text{ (and G is a star graph)} \\
&x_j + y_i \ge 1 && |\Delta(j)| = n, \quad |\Delta(i)| = 2 \\
&x_j = \sum_{i\in C} y_i && \Delta(j) = C \\
&x_i = x_j && \Delta(i) = \Delta(j) \\
&y_k + x_j \le 1 + x_i && k \in \Delta(i) \subseteq \Delta(j) \\
&y_i + y_j + y_k + y_\ell \le 2 && \{\{i,j\},\{j,k\},\{k,\ell\}\} \in E, \quad |\Delta(j)| = |\Delta(k)| = 3 \\
&y_i + y_\ell \le x_j + x_k && \{\{i,j\},\{j,k\},\{k,\ell\}\} \in E, \quad |\Delta(j)| = |\Delta(k)| = 3
\end{aligned}
\tag{6.25}
$$

In this model, we did not list the constraints that are implied by one of the other constraints. For example, Constraint 6.11 and 6.17 are created by making Constraint 6.3 and 6.4 stronger, respectively. We realized that $\sum_{i\in S} y_i \le 1$ if $x_j = 1$ and $S \subseteq \Delta(j)$. Thus, we added x_j with a coefficient to Constraint 6.3 and 6.4 to create Constraints 6.11 and 6.17. This direct relationship enabled us to easily remove Constraints 6.3 and 6.4 since they are implied by the new ones.

However, there might be more complex relationships between two or more constraints of the finalized model, and there might other constraints that are implied by one or more other constraints of the finalized model.

One important thing is that the constraints in the finalized model are given with the same order as listed in this chapter. They are all created to cut a fractional solution off that is found to be optimal to the LP relaxation with the previous set of constraints. For example, there is at least one fractional solution (see Figure 6.27) that is cut by Constraint 6.24, $y_i+y_\ell \le x_j+x_k, \quad \{\{i,j\},\{j,k\},\{k,\ell\}\} \in E, \quad |\Delta(j)| = |\Delta(k)| = 3$, and that solution is found to be optimal with the previous set of constraints (all

other constraints). Thus, we can say that Constraint 6.24 is not a redundant constraint in the finalized model. However, the same does not apply for the other constraints. It is clear that a constraint cannot be implied by the previous set of constraints, but there is a chance that it is implied by the set of constraints that are added after the constraint (or implied by the combination of the previous set of constraints and the set of constraints that are added after the constraint).

Additionally, while adding the valid inequalities, we found a couple of other valid inequalities that are not listed here. Some of these are eliminated because they are exponentially many (note that all of the constraints of our finalized model are polynomially many) or because they are too specific. Now, we will go over these constraints.

Proposition 6.23. $\sum_{i\in S_2} x_i + |S_1| - 1 \geq \sum_{k\in S_1} y_k + (|S_1| - 2)x_j, \quad \forall j \in V$ where $S_1 \cup S_2 = \Delta(j)$, $S_1 \cap S_2 = \emptyset$, $j \in S_1$ and $|S_1| \geq 2$ is a valid inequality for an arbitrary graph for the MinWPNSP.

Proof.

Case 1. $x_j = 1$. Then we have $\sum_{k\in S_1} y_k \leq 1$ since since $x_j = 1$ implies $\sum_{k\in\Delta(j)} y_k = 1$ and we have $S_1 \subset \Delta(j)$. It is, then, implied that $\sum_{k\in S_1} y_k + (|S_1| - 2)x_j \leq |S_1| - 1$. In this case, the constraint will always be valid for all values of $\sum_{i\in S_2} x_i$.

Case 2. $x_j = 0$ and $\sum_{k\in S_1} y_k \leq |S_1| - 1$. Then, similar to the Case 1, we have $\sum_{k\in S_1} y_k + (|S_1| - 2)x_j \leq |S_1| - 1$ and always satisfy the constraint.

Case 3. $x_j = 0$ and $\sum_{k\in S_1} y_k = |S_1|$. In this case, it is clear that $y_k = 1$ for all $k \in S_1$. Since $S_1 \subset \Delta(j)$, we have that $j \in \Delta(k)$ for all $k \in S_1$ and $\sum_{\ell\in\Delta(k)} y_\ell \geq 2$, implying $x_k = 0$ for all $k \in S_1$.

Constraint 4.4 implies that $\sum_{i\in\Delta(j)} x_i = \sum_{k\in S_1} x_k + \sum_{i\in S_2} x_i \geq 1$. In our case, we have $\sum_{k\in S_1} x_k = 0$, thus constraint will imply $\sum_{i\in S_2} x_i \geq 1$. Overall, left-hand side of the constraint will be $\sum_{i\in S_2} x_i + |S_1| - 1 \geq |S_1|$ and the right-hand side will be equal to $|S_1|$, which is always feasible.

□

This constraint is a generalized version of Constraint 6.11. For a vertex j, if S_2 is

selected to be an empty set, then the constraint in Proposition 6.23 will be the same of Constraint 6.11 for vertex j. Thus, it implies Constraint 6.11. Additionally, the reader may observe that the number of constraints in Proposition 6.23 will be exponentially many. For a vertex j, there will be $\mathcal{O}(2^{|\Delta(j)|})$ many constraints in Proposition 6.23

Another constraint class with exponentially many constraints is given in the following proposition.

Proposition 6.24. $|S_1| \geq \sum_{k \in S_1} y_k + (|S_1| - 1)x_j, \quad \forall j \in V$ where $S_1 \subseteq \Delta(j)$ is a valid inequality for an arbitrary graph for the MinWPNSP.

Proof.

Case 1. $x_j = 1$. Then we have $\sum_{k \in S_1} y_k \leq 1$ since since $x_j = 1$ implies $\sum_{k \in \Delta(j)} y_k = 1$ and we have $S_1 \subseteq \Delta(j)$. It is, then, implied that $\sum_{k \in S_1} y_k + (|S_1| - 1)x_j \leq |S_1|$.

Case 2. $x_j = 0$. Then, the constraint is again satisfied since all decision variables are bounded to be at most 1.

□

This constraint set, similar to the constraint in Proposition 6.23 will have $\mathcal{O}(2^{|\Delta(j)|})$ many constraints for a vertex j.

The constraint in Proposition 6.24 can be written for all subsets of $\Delta(j)$ as S_1. If $S_1 = \emptyset$, it implies $x_j \geq 0$. Else if $|S_1| = 1$, say $S_1 = \{i\}$, then it implies $y_i \leq 1$. Else if $|S_1| = 2$, say $S_1 = \{i, k\}$, then it implies $x_j + y_i + y_k \leq 2$ which is also implied by Constraint 4.6. Else if $|S_1| = \Delta(j)$, $|\Delta(j)| \geq \sum_{k \in \Delta(j)} y_k + (|\Delta(j)| - 1)x_j$ would be a valid constraint for the problem but will be implied by Constraint 6.11 for vertex j. For the rest of the cardinalities of S_1, the constraint might not be implied by the current model.

The next constraint that we found and did not add to the model is given below.

Proposition 6.25. $\sum_{i \in S_1} y_i + \sum_{i \in S_2} x_i \leq |C| + 1$ where C is a clique, $j, t \in C$, $k \in \Delta(j) - C$, $S_1 = C \cup \{k\}$ and $S_2 = S_1 - \{t\}$, is a valid inequality for an arbitrary graph for the MinWPNSP.

Proof. In this proposition, we are given with a clique, C, which includes two distinct vertices as j and t, and k is a vertex that is adjacent to j but do not belong to C. Additionally, note that we have $|S_1| = C + 1$ and $|S_2| = C$. Because of these three vertices, we naturally have $|S_1| \geq 3$.

First of all, let us assume that we have $\sum_{i \in S_1} y_i \geq 3$, then we have $\sum_{i \in C} y_i \geq 2$ since $S_1 = C \cup \{k\}$. In this case, $\sum_{i \in C} x_i = 0$ since at least two vertices from C is selected to be in the set S, making all of the vertices in the clique as not S-perfect vertices. Overall, if $\sum_{i \in S_1} y_i \geq 3$, then $\sum_{i \in S_2} x_i = x_k$, or $\sum_{i \in S_2} x_i \leq 1$.

Case 1. $\sum_{i \in S_1} y_i = |S_1|$. Then, we have $\sum_{i \in S_2} x_i = x_k$. In this case, $x_k = 0$ because we have $y_k = y_j = 1$, making $\sum_{i \in \Delta(k)} y_i \geq 2$. Overall, $\sum_{i \in S_1} y_i + \sum_{i \in S_2} x_i = |S_1| + 0 = |S_1|$ implies and the constraint would be satisfied.

Case 2. $3 \leq \sum_{i \in S_1} y_i < |S_1|$. First of all, note that in this case, we have $|S_1| \geq 4$. Similar to the Case 1, we have $\sum_{i \in S_2} x_i \leq 1$ since $\sum_{i \in S_1} y_i \geq 3$. Thus, $\sum_{i \in S_1} y_i + \sum_{i \in S_2} x_i \leq |S_1|$ is implied and the constraint, then, will be satisfied.

Case 3. $\sum_{i \in S_1} y_i = 2$. Then, $x_j = 0$ since there exists at least two vertex in $\Delta(j)$ which are selected to be in S. Thus, we have $\sum_{i \in S_2} x_i \leq |S_2| - 1 = |C| - 1$. Overall, $\sum_{i \in S_1} y_i + \sum_{i \in S_2} x_i \leq 2 + |C| - 1$, which satisfies the constraint.

Case 4. $\sum_{i \in S_1} y_i \leq 1$. We naturally have $\sum_{i \in S_2} x_i \leq |S_2| = |C|$ and the constraint is always satisfied with $\sum_{i \in S_1} y_i + \sum_{i \in S_2} x_i \leq 1 + |C|$.

□

When $|S_1| = 3$, the constraint in Proposition 6.25 will be equal to Constraint 6.7. We created constraint in Proposition 6.25 to generalize Constraint 6.7 for more cases, since Constraint 6.7 is seen to be effective in densely connected graphs. The effects of the constraints will be discussed in more detail in Chapter 7.

6.1 Perfect Neighborhood Set Polytope

While working on the additional valid inequalities, we tried to characterize the convex hull of the feasible region of the IP formulation which is called as the perfect

neighborhood set polytope for specific graph classes. We will now go over these in this section.

6.1.1 Star Graph

A star graph is a tree with one vertex having the degree of $|V| - 1$ and the rest of the vertices having 1. Consider the graph $G = (V, E)$ where $V = \{0, 1, 2, \ldots, n\}$ and $E = \{\{0, 1\}, \{0, 2\}, \ldots, \{0, n\}\}$. This graph defines the star graph in Figure 6.28. In the literature, notation of S_n and "n-star" are used to represent a star graph where the order, $|V|$, of the star graph is n. In our example in the figure, the order is $n + 1$.

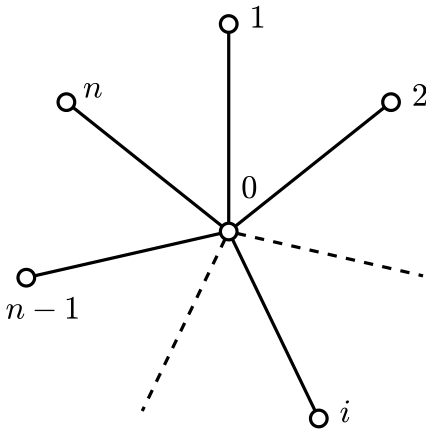


Figure 6.28: A star graph with order of $n + 1$.

To possibly define the perfect neighborhood set polytope of a star graph, we first listed all of the PN sets on a star graph as given in Figure 6.28. Overall, we have three types of PN sets.

1. $S = \{0\}$, implying S-perfect $= V$.
2. $S = \{i\}$, where $i \in V - \{0\}$, implying S-perfect $= \{0, i\}$.
3. $S = \{1, 2, \ldots, n\}$, implying S-perfect $= \{1, 2, \ldots, n\}$.

In the first solution, we select only vertex 0 to be in S. In this case, all of the vertices of the star graph are perfect. Thus, S-perfect vertices define a dominating set where each vertex dominates itself.

In the second PN sets, we select one of the vertices with a degree of 1, let it be denoted by $i \in V - \{0\}$. In such a scenario, vertices 0 and i are identified as perfect vertices. Since vertex 0 is adjacent to the whole graph, S-perfect vertices define a dominating set. Thus, this scenario also defines a feasible solution.

In the third solution, we select all of the vertices except vertex 0 to be in S, which makes all of the vertices except vertex 0 perfect. The only vertex that is not perfect, i.e. vertex 0, is dominated by one of the other vertices.

First and third solutions are unique. We have, however, n different second solutions for each $i \in V - \{0\}$. Overall, for a star graph of order $n+1$, we have $n+2$ different PN sets.

After identifying these solutions, we created the incidence vectors of all solutions. For a star graph of order $n+1$, we have two different sets of decision variables, y and x, each is defined for each vertex of the star graph. Thus, each incidence vector is in $2n+2$-dimensional space. Overall, Equation 6.26 is the set of feasible solutions to the IP for a star graph of order $n+1$, where vertex 0 is the vertex having degree $|V|-1$.

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_n \\ x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \in \left\{ \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 1 \\ 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \\ 1 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \\ \vdots \\ 0 \\ 1 \\ 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix}, \ldots, \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 1 \\ 1 \\ 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 1 \\ \vdots \\ 1 \\ 0 \\ 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \right\} \tag{6.26}$$

First and last vectors of the set of feasible solutions in Equation 6.26 are for the first and the third solution listed above, respectively. In between, we have n different vectors for each solution of the second item.

Then, we defined the convex hull of the perfect neighborhood set polytope, $conv(S)$,

by adding additional variables, α_j for each solution where $j \in \{0, 1, 2, \ldots, n+1\}$.

$$conv(S) = \left\{ \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_n \\ x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} ; \alpha_0 \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 1 \\ 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} + \alpha_1 \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \\ 1 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} + \alpha_2 \begin{bmatrix} 0 \\ 0 \\ 1 \\ \vdots \\ 0 \\ 1 \\ 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix} + \cdots + \alpha_n \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 1 \\ 1 \\ 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix} + \alpha_{n+1} \begin{bmatrix} 0 \\ 1 \\ 1 \\ \vdots \\ 1 \\ 0 \\ 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \right\} \tag{6.27}$$

where

$$\alpha_0 + \alpha_1 + \cdots + \alpha_{n+1} = 1, \tag{6.28}$$

$$\alpha_j \geq 0, \quad \forall j \in \{0, 1, \ldots, n+1\} \tag{6.29}$$

Here, Equation 6.27 defines a total of $2n + 2$ equations for each decision variable of the PN sets. These are listed below:

$$y_0 = \alpha_0 \tag{6.30}$$

$$y_1 = \alpha_1 + \alpha_{n+1} \tag{6.31}$$

$$y_2 = \alpha_2 + \alpha_{n+1} \tag{6.32}$$

$$\vdots$$

$$y_i = \alpha_i + \alpha_{n+1}, \quad \forall i \in \{1, \ldots, n\} \tag{6.33}$$

$$\vdots$$

$$y_n = \alpha_n + \alpha_{n+1} \tag{6.34}$$

$$x_0 = \alpha_0 + \alpha_1 + \cdots + \alpha_n \tag{6.35}$$

$$x_1 = \alpha_0 + \alpha_1 + \alpha_{n+1} \tag{6.36}$$

$$x_2 = \alpha_0 + \alpha_2 + \alpha_{n+1} \tag{6.37}$$

$$\vdots$$

$$x_i = \alpha_0 + \alpha_i + \alpha_{n+1}, \quad \forall i \in \{1, \ldots, n\} \tag{6.38}$$

$$\vdots$$

$$x_n = \alpha_0 + \alpha_n + \alpha_{n+1} \tag{6.39}$$

Then, we tried to reduce the number of variables of the convex hull definition by

defining the vector of α in terms of vectors of y and x. By Equation 6.30, we have that $\alpha_0 = y_0$. Moreover, Equation 6.33 implies that $\alpha_i = y_i - \alpha_{n+1}$ for each $i \in \{1, 2, \ldots, n\}$. Combining these two results in Equation 6.35, we have the following equation:

$$x_0 = y_0 + (y_1 - \alpha_{n+1}) + (y_2 - \alpha_{n+1}) \cdots + (y_n - \alpha_{n+1})$$

By leaving α_{n+1} in one side, we have the equation as:

$$\alpha_{n+1} = \frac{y_0 + y_1 + \cdots + y_n - x_0}{n} \tag{6.40}$$

Moreover, if we add the result in Equation 6.40 into Equation 6.33, we have:

$$\alpha_i = y_i - \frac{y_0 + y_1 + \cdots + y_n - x_0}{n}, \quad \forall i \in \{1, \ldots, n\} \tag{6.41}$$

Combining the results, we were able to define α in terms of the rest of the variables, y and x:

$$\alpha_0 = y_0 \tag{6.42}$$

$$\alpha_1 = y_1 - \frac{y_0 + y_1 + \cdots + y_n - x_0}{n} \tag{6.43}$$

$$\alpha_2 = y_2 - \frac{y_0 + y_1 + \cdots + y_n - x_0}{n} \tag{6.44}$$

$$\vdots$$

$$\alpha_n = y_n - \frac{y_0 + y_1 + \cdots + y_n - x_0}{n} \tag{6.45}$$

$$\alpha_{n+1} = \frac{y_0 + y_1 + \cdots + y_n - x_0}{n} \tag{6.46}$$

Including that, we have $y_i = \alpha_i + \alpha_{n+1}$ by Equation 6.33 and $x_i = \alpha_0 + \alpha_i + \alpha_{n+1}$ by Equation 6.38. Inserting 6.33 into the other, we have $x_i = \alpha_0 + y_i$ for each $i \in \{1, \ldots, n\}$. By Equation 6.42, we have $\alpha_0 = y_0$ and the combination of these results gives us the following equation:

$$x_i = y_0 + y_i, \quad \forall i \in \{1, \ldots, n\} \tag{6.47}$$

This equation is directly implied by our model with Equation 6.6. In our star graph, i would be a pendant vertex and $\{i, 0\} \in E$.

Moreover, Equation 6.28 implies the sum of α values are 1. We already have $\alpha_0 + \alpha_1 + \cdots + \alpha_n = x_0$ and $\alpha_{n+1} = (y_0 + y_1 + \cdots + y_n - x_0)/n$. By summing those equations, we have;

$$\alpha_0 + \alpha_1 + \cdots + \alpha_{n+1} = x_0 + \frac{y_0 + y_1 + \cdots + y_n - x_0}{n} = 1$$

and

$$nx_0 + (y_0 + y_1 + \cdots + y_n - x_0) = n$$

Overall, this equation gives us the following;

$$(n-1)x_0 + \sum_{j \in V} y_j = n \tag{6.48}$$

This, again, is also implied by our model with Equation 6.18. Note that in our example, we have $|V| = n + 1$.

Lastly, convex hull implies Equation 6.29. Each of the α values must be nonnegative.

$$\begin{aligned}
\alpha_0 &= y_0 \geq 0 \\
\alpha_1 &= y_1 - \frac{y_0 + y_1 + \cdots + y_n - x_0}{n} \geq 0 \\
\alpha_2 &= y_2 - \frac{y_0 + y_1 + \cdots + y_n - x_0}{n} \geq 0 \\
&\vdots \\
\alpha_n &= y_n - \frac{y_0 + y_1 + \cdots + y_n - x_0}{n} \geq 0
\end{aligned}$$

$$\alpha_{n+1} = \frac{y_0 + y_1 + \cdots + y_n - x_0}{n} \geq 0$$

By the first and the last one of nonnegativity constraints respectively, we have

$$y_0 \geq 0 \tag{6.49}$$

$$x_0 \leq \sum_{j \in V} y_j \tag{6.50}$$

Equation 6.49 is directly implied by our model with the LP relaxation of Equation 4.8 in Equation 6.1. Moreover, Equation 6.50 is implied by Equation 4.5 for $j = 0$ since $\Delta(0) = V$.

Additionally, the nonnegativity constraints corresponding to α values from α_1 to α_n implies that

$$y_i - \frac{y_0 + y_1 + \cdots + y_n - x_0}{n} \geq 0, \quad \forall i \in \{1, \ldots, n\}$$

and

$$n y_i + x_0 \geq \sum_{j \in V} y_j, \quad \forall i \in \{1, \ldots, n\} \tag{6.51}$$

Proposition 6.26. Equation 6.51 is implied by our model with additional valid inequalities.

Proof. Equation 6.19 of the model implies that $x_j + y_i \geq 1, \quad |\Delta(j)| = n, \quad |\Delta(i)| = 2$ and note that for this equation we have $|V| = n$, i.e. it is valid when $|\Delta(j)| = |V|$. Thus, this is written for our star graph representation when $j = 0$ and $i \in \{1, \ldots, n\}$. Overall, model implies that

$$x_0 + y_i \geq 1, \quad \forall i \in \{1, \ldots, n\}$$

and by multiplying each side by n and subtracting $(n-1)x_0$ from both of the sides, model implies

$$ny_i + x_0 \geq n - (n-1)x_0, \quad \forall i \in \{1, \ldots, n\} \tag{6.52}$$

Moreover, Equation 6.18 of the model implies that $(|\Delta(j)| - 2)x_j + \sum_{i \in \Delta(j)} y_i = |\Delta(j)| - 1, \quad \forall j \in Sup$ if the graph is a star graph. For our case, $0 \in Sup$, $|\Delta(0)| = n + 1$, and $\Delta(0) = V$. Overall, model implies the following equation

$$(n-1)x_0 + \sum_{j \in V} y_j = n$$

by subtracting $(n-1)x_0$ from both sides, we imply

$$\sum_{j \in V} y_j = n - (n-1)x_0 \tag{6.53}$$

By combining Equation 6.52 and 6.53, the model implies

$$ny_i + x_0 \geq n - (n-1)x_0 = \sum_{j \in V} y_j, \quad \forall i \in \{1, \ldots, n\}$$

and finally,

$$ny_i + x_0 \geq \sum_{j \in V} y_j, \quad \forall i \in \{1, \ldots, n\}$$

□

With the reduction of α values, the convex hull of the perfect neighborhood set problem on star graph converts into Equation 6.54, where each of the constraints on the right hand side proved to be implied by our model with additional valid inequalities.

Let us assume that P is the polytope of our model with all additional valid inequalities and $P1$ is the polytope that implies the convex hull in (6.54), which is the union of the model with valid inequalities 6.6, 6.18 and 6.19.

Since all of the valid inequalities are defined to be valid for the perfect neighborhood set polytope of a star graph, we have that

$$conv(S) \subseteq P$$

$$conv(S) = \left\{ \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_n \\ x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} : \begin{array}{lll} \text{subject to} & & \\ y_0 \geq 0 & & \\ x_0 \leq \sum_{j \in V} y_j & & \\ x_i = y_0 + y_i & & \forall i \in \{1, \ldots, n\} \\ n y_i + x_0 \geq \sum_{j \in V} y_j & & \forall i \in \{1, \ldots, n\} \\ (n-1)x_0 + \sum_{j \in V} y_j = n & & \\ x_j,\ y_j \ \text{urs} & & \forall j \in V \end{array} \right\} \tag{6.54}$$

Additionally, P includes all of the constraints of $P1$ and an addition of the rest of the valid inequalities, we have

$$conv(S) \subseteq P \subseteq P1$$

Moreover, we have proved that all of the constraints of $conv(S)$ are defined by $P1$. Thus,

$$P1 \subseteq conv(S)$$

implying

$$conv(S) = P = P1 \tag{6.55}$$

for a star graph.

Proposition 6.27. Dimension of the perfect neighborhood set polytope in star graphs, $dim(P)$, is $|V|$.

Proof. The list of solutions for a star graph (with $|V| = n+1$ where vertex 0 being the support vertex) was given in 6.26. It is known that $x^1, x^2, \ldots, x^k \in \mathbb{R}^n$ are affinely

independent if the only solution of the system $\sum_{i=1,\dots,k} \alpha_i x^i = 0$ and $\sum_{i=1,\dots,k} \alpha_i = 0$ is $\alpha_i = 0$ for all $i = 1, \dots, k$. Let us create such a system of equations for the list of the solutions of a star graph.

$$\alpha_0 \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 1 \\ 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} + \alpha_1 \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \\ 1 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} + \alpha_2 \begin{bmatrix} 0 \\ 0 \\ 1 \\ \vdots \\ 0 \\ 1 \\ 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix} + \cdots + \alpha_n \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 1 \\ 1 \\ 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix} + \alpha_{n+1} \begin{bmatrix} 0 \\ 1 \\ 1 \\ \vdots \\ 1 \\ 0 \\ 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} = 0 \tag{6.56}$$

$$\alpha_0 + \alpha_1 + \cdots + \alpha_{n+1} = 0 \tag{6.57}$$

By solving the first $|V|$ equations of 6.56, we have

$$\alpha_0 = 0 \tag{6.58}$$

$$\alpha_1 + \alpha_{n+1} = 0 \tag{6.59}$$

$$\alpha_2 + \alpha_{n+1} = 0 \tag{6.60}$$

$$\vdots$$

$$\alpha_n + \alpha_{n+1} = 0 \tag{6.61}$$

These equations imply that $\alpha_0 = 0$ and $\alpha_1 = \alpha_2 = \cdots = \alpha_n = -\alpha_{n+1}$. Additionally, we have $\alpha_0 + \alpha_1 + \cdots + \alpha_{n+1} = 0$ by 6.57 and $\alpha_0 + \alpha_1 + \cdots + \alpha_n = 0$ by the $|V| + 1$th equation of 6.56, implying $\alpha_{n+1} = 0$. Then, it is implied that $\alpha_1 = \alpha_2 = \cdots = \alpha_n = 0$, which implies that list of the solutions of the perfect neighborhood set problem on a star graph are affinely independent of each other. Then, we can add that $dim(P) \geq |V|$ since the number of affinely independent vectors of the solution set is $|V| + 1$.

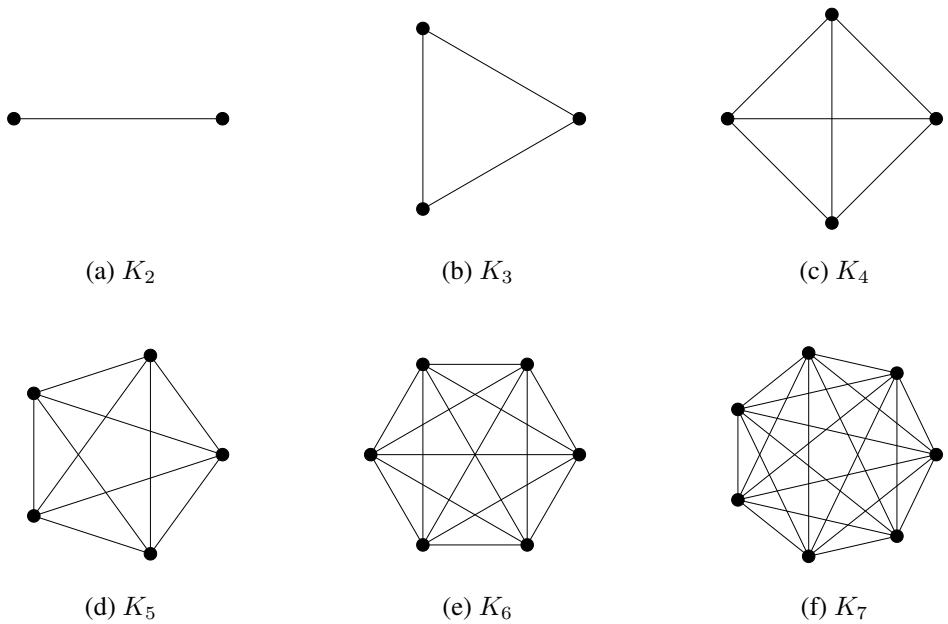
Additionally, the system of equations of the convex hull of the star graph, 6.54, imply two types of equality constraints as $(n-1)x_0 + \sum_{j \in V} y_j = n$ and $x_i = y_0 + y_i$ for all $i \in \{1, \ldots, n\}$. Since each of these equality constraints include a unique x value with a nonzero multiplier, we can add that the rank of the equality constraints of the star graph, $rank(A^{=})$, is at least $|V|$ (assuming that the polytope of the star graph is in the form $P = \left\{ x \in \mathbb{R}^{2|V|} | Ax \leq b \right\}$ and $A^{=}$ is the submatrix of the A including all the equality constraints).

It is known that if $P \neq \emptyset$ and $P \subseteq \mathbb{R}^{2|V|}$, then $dim(P) + rank(A^{=}) = 2|V|$. Since we have already shown that $dim(P) \geq |V|$ and $rank(A^{=}) \geq |V|$, we conclude that $dim(P) = |V|$.

□

6.1.2 Complete Graph

A complete graph, denoted by K_n, is a graph that includes one edge for any pair of vertices. A graph, $G = (V, E)$ where $V = \{1, 2, \ldots, n\}$ and $E = \{\{i, j\} | i \in V, j \in V, i \neq j\}$ is a complete graph denoted as K_n.



(a) K_2 (b) K_3 (c) K_4

(d) K_5 (e) K_6 (f) K_7

Figure 6.29: Illustration of complete graphs from K_2 to K_7.

Similar to Section 6.1.1, we started by listing of all the PN sets in a complete graph of order n. For a complete graph, there is only one type of a solution: one of the vertices is selected to be in the set of S, which implies $S - perfect = V$. In a case that more than one vertex is selected to be in S, we have $S - perfect = \emptyset$ since each of the vertices of the graph includes two or more vertices from S in its closed neighborhood. Overall, Equation 6.62 includes the list of all feasible solution of perfect neighborhood set problem on a complete graph of order n.

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \\ x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} \in \left\{ \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 1 \\ 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \\ 1 \\ 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \\ \vdots \\ 0 \\ 1 \\ 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}, \ldots, \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 1 \\ 1 \\ 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \right\} \tag{6.62}$$

With the addition of α values, the convex hull of the MinWPNSP on complete graphs is defined as in Equation 6.63.

$$conv(K_n) = \left\{ \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \\ x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} ; \alpha_1 \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 1 \\ 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} + \alpha_2 \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \\ 1 \\ 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} + \alpha_3 \begin{bmatrix} 0 \\ 0 \\ 1 \\ \vdots \\ 0 \\ 1 \\ 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} + \cdots + \alpha_n \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 1 \\ 1 \\ 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \right\} \tag{6.63}$$

where

$$\alpha_1 + \alpha_2 + \cdots + \alpha_n = 1, \tag{6.64}$$

$$\alpha_i \geq 0, \quad \forall i \in \{1, 2, \ldots, n\} \tag{6.65}$$

More explicitly, Equation 6.63 defines the following equations

$$y_j = \alpha_j, \quad \forall j \in V \tag{6.66}$$

$$x_j = \alpha_1 + \alpha_2 + \cdots + \alpha_n, \quad \forall j \in V \tag{6.67}$$

In this case, we directly define the vector of α by y and x values as $\alpha_i = y_i, \quad \forall i \in \{1, 2, \ldots, n\}$. Additionally, by Equation 6.64 and 6.67, we have $x_j = 1, \quad \forall j \in V$. By combining the results of Equation 6.65 and 6.66, we have $y_j \geq 0, \quad \forall j \in V$. Finally, if we sum all of Equation 6.66, we have $\sum_{j \in V} y_j = \alpha_1 + \alpha_2 + \cdots + \alpha_n = 1$. By eliminating the α values and combining these results, we can characterize the perfect neighborhood set polytope in complete graphs as in Equation 6.68.

$$conv(K_n) = \left\{ \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} : \begin{array}{lll} \text{subject to} & & \\ y_j \geq 0 & & \forall j \in V \\ x_j = 1 & & \forall j \in V \\ \sum_{i \in V} y_i = 1 & & \end{array} \right\} \tag{6.68}$$

Proposition 6.28. Addition of Constraints 6.2 and 6.20 to the LP relaxation of base model defines the pefect neighborhood set polytope in complete graphs.

Proof. The relaxed version of the binary constraint, given in Constraint 6.1 directly implies the first constraint of the convex hull description above; $y_j \geq 0, \quad \forall j \in V$.

Additionally, Constraint 6.20 is written for each maximal clique, C, of a graph. In a complete graph, there is only one maximal clique, say C_1, as $C_1 = V$. Moreover, since each vertex is adjacent to all the other vertices of the graph, we have

$\Delta(j) = V = C_1, \quad \forall j \in V$. Thus, Constraint 6.20 implies that $x_j = \sum_{i \in C_1} y_i = \sum_{i \in V} y_i, \quad \forall j \in V$.

Constraint 6.2 implies that at least one vertex should be selected to be in the set of S from the double neighborhood of each vertex of the graph. In a complete graph, we have $\Delta(j) = \Delta^2(j) = V, \quad \forall j \in V$. Thus, all of these constraints of 6.2 define the same which is $\sum_{i \in V} y_i \geq 1$.

By combining the results, we have $x_j = \sum_{i \in V} y_i \geq 1, \quad \forall j \in V$. Since the relaxed version of the binary constraint, Constraint 6.1, also implies that $x_j \leq 1, \quad \forall j \in V$, we have $x_j = \sum_{i \in V} y_i = 1, \quad \forall j \in V$, which implies the second and the third constraint of the convex hull description of a complete graph given in Equation 6.68. □

Let P_{K_n} define the polytope of the LP relaxation of the base model with the addition of Constraints 6.2 and 6.20. We have proved that

$$P_{K_n} \subseteq conv(K_n)$$

Since all of the valid inequalities are defined to be valid for a polytope of a graph, we have

$$conv(K_n) \subseteq P \subseteq P_{K_n}$$

Combination of the results give us

$$conv(K_n) = P = P_{K_n} \tag{6.69}$$

for a complete graph.

Proposition 6.29. Dimension of the perfect neighborhood set polytope in complete graphs, $dim(P_{K_n})$, is $|V| - 1$.

Proof. The set of feasible solution of a complete graph, as given in Equation 6.62 are all affinely independent of each other since each includes a unique y value with a nonzero coefficient. Since we found $|V|$ affinely independent vectors in the polytope, we can add that $dim(P_{K_n}) \geq |V| - 1$.

Additionally, we can easily see that $rank(A^{=}_{K_n}) \geq |V| + 1$ from Equation 6.68. All of those $x_j = 1$ for all $j \in V$ constraints have a unique x value with a nonzero coefficient and have zero coefficient on y_j values for all $j \in V$. Then, we can add that $|V|$ many constraints of x values being equal to 1 and the one constraint as all of the y values should sum to one are linearly independent of each other. That implies that $rank(A^{=}_{K_n}) \geq |V| + 1$, as total number of linearly independent constraints of the convex hull description of the polytope.

By using the fact that if $P \neq \emptyset$ and $P \subseteq \mathbb{R}^{2|V|}$, then $dim(P) + rank(A^{=}) = 2|V|$, we conclude that $dim(P_{K_n}) = |V| - 1$. □

CHAPTER 7

EFFECTS OF THE VALID INEQUALITIES

The runs in this chapter are taken on a computer with Intel Core i7-4770S CPU @3.10GHz (8 CPUs) and 16.00GB RAM. We use CPLEX 12.9 through C++ API (Visual Studio 2019, v142) to solve the mathematical models.

We will first go over the effects of the additional valid inequalities for different graph classes. To do that, we created a couple of models. All of those models share the same objective function; minimization of the weighted case of the problem, and have the base constraints that are given at the beginning of Chapter 6 in (6.1). Note that the binary constraints are relaxed in (6.1).

$$
\begin{aligned}
\text{minimize} \quad & \sum_{j \in V} (w_j y_j + v_j x_j) & & \\
\text{subject to} \quad & \sum_{i \in \Delta(j)} x_i \geq 1 & & \forall j \in V \\
& x_j \leq \sum_{i \in \Delta(j)} y_i & & \forall j \in V \\
& x_j + y_l + y_k \leq 2 & & \forall j \in V, \quad \forall \{k, l\} \in \Delta(j), \quad k \neq l \\
& x_j + \sum_{i \in \Delta(j), i \neq k} y_i \geq y_k & & \forall j \in V, \quad \forall k \in \Delta(j) \\
& 0 \leq x_j,\ y_j \leq 1, & & \forall j \in V
\end{aligned}
$$

We created a set of 10 different objective functions. Half of them weights only the vertices that are selected to be in the set S, and the second half of them weights for both sets S and $S - perfect$. Each of these 5 objective functions includes a different range to randomize the weights in between. These are $[1, 1]$, $[-1, -1]$, $[0, 1]$, $[-1, 0]$, and $[-1, 1]$.

For example, we discussed the number $\theta(G)$ ($\Theta(G)$) for MinCPNSP (MaxCPNSP), in which the cardinality of the set of S-perfect vertices are ignored. In our context, this objective function represents the one where we weight only the vertices that are selected to be in the set S, thus $v_j = 0$ for all $j \in V$, and the range we randomize the parameter w will be in $[1, 1]$ ($[-1, -1]$), i.e., w will be a vector of all (minus) ones. In other words, the first two ranges of the weights, $[1, 1]$ and $[-1, -1]$, does not randomize the weights and assign the value of 1 and -1 to all of them, respectively. $[0, 1]$ and $[-1, 0]$ represent the ones that we minimize and maximize the weighted cardinality among all PN sets of G, respectively. Lastly, $[-1, 1]$ creates an objective function where a mixture of minimization and maximization applies.

7.1 Effects of the Valid Inequalities in Trees

We first test our valid inequalities on trees. By using Algorithm 4, we have created 1000 trees for each number of vertices of the set $|V| \in \{5, 10, 25, 50, 75, 100\}$. Then, we have created the corresponding weight values for these trees individually and solved them with the base constraints that are given in (6.1). The reader can find the percentage values of these solutions in which the LP relaxation gives an integral solution in Table 7.1. In this table, 1 and -1 are used to represent the ranges of $[1, 1]$ and $[-1, -1]$, respectively, and a value is written with bold if it is exactly 0 or 100 percent.

Table 7.1: Percentage of the integral solutions of 1000 trees with the base LP relaxation model.

$\|V\|$	weight on both y and x						weight on y					
	1	-1	$[0, 1]$	$[-1, 0]$	$[-1, 1]$	Avg.	1	-1	$[0, 1]$	$[-1, 0]$	$[-1, 1]$	Avg.
5	**0.0%**	9.3%	10.1%	41.0%	36.0%	19.3%	5.5%	9.3%	37.3%	8.1%	41.6%	20.4%
10	6.7%	12.8%	11.2%	20.4%	14.5%	13.1%	10.3%	0.6%	15.3%	0.9%	17.5%	8.9%
25	0.3%	0.3%	0.3%	1.4%	1.2%	0.7%	0.1%	**0.0%**	0.3%	**0.0%**	1.5%	0.4%
50	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	0.1%	0.0%
75	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**
100	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**
Avg.	1.2%	3.7%	3.6%	10.5%	8.6%	5.5%	2.7%	1.7%	8.8%	1.5%	10.1%	4.9%

Then, to understand the cumulative effect of the valid inequalities, we have solved the same trees with the same objective function values with the combination of the base model with all valid inequalities that are given in the finalized model in Chapter 6. The percentage values of these solutions are given in Table 7.2.

Table 7.2: Percentage of the integral solutions of 1000 trees with the finalized model.

$\|V\|$	weight on both y and x						weight on y					
	1	-1	$[0,1]$	$[-1,0]$	$[-1,1]$	Avg.	1	-1	$[0,1]$	$[-1,0]$	$[-1,1]$	Avg.
5	**100.0%**	**100.0%**	98.4%	**100.0%**	99.0%	99.5%	**100.0%**	**100.0%**	**100.0%**	**100.0%**	99.8%	100.0%
10	89.4%	**100.0%**	94.6%	98.9%	90.5%	94.7%	99.8%	89.7%	98.2%	81.9%	92.7%	92.5%
25	79.6%	99.9%	79.7%	96.8%	72.7%	85.7%	99.5%	71.4%	96.6%	55.3%	75.4%	79.6%
50	57.8%	99.8%	61.8%	94.4%	54.1%	73.6%	93.0%	48.1%	94.6%	28.6%	55.9%	64.0%
75	44.8%	99.4%	48.2%	88.5%	35.7%	63.3%	93.6%	34.4%	90.7%	14.9%	40.5%	54.8%
100	37.7%	99.0%	37.6%	83.7%	28.0%	57.2%	90.0%	21.5%	87.0%	7.2%	31.5%	47.4%
Avg.	68.2%	99.7%	70.1%	93.7%	63.3%	79.0%	96.0%	60.9%	94.5%	48.0%	66.0%	73.1%

If the solution of the LP relaxation with the additional valid inequalities is integral, then the solution would also be the optimal solution of the corresponding IP model. Thus, by the help of the valid inequalities, the average percentage of the solutions which can be found with the LP relaxation increases from 5.5% and 4.9% to 79.0% and 73.1% with the valid inequalities, for the weights on both y and x and the weight on only y, respectively.

It is found that valid inequalities are significantly helpful on the instances where weights are assigned as -1 for both y and x, because the average percentage increases from 3.7% to 99.7%. Even for a large tree with $|V| = 100$, the average percentage of the integral solutions is found as 99.0%. Another significant improvement with the valid inequalities occurs for the problems with weights in between $[0, 1]$ for only y. The average percentage of the integral solutions for these problems increases from 8.8% to 94.5%. Lastly, a similar improvement occurs on the range $[-1, 0]$ for both y and x where the percent increases from 10.5% to 93.7%.

With the addition of the valid inequalities, the number $\theta(T)$ can be found by solving an LP relaxation with 96.0% chance for our instances. Even for a large tree with $|V| = 100$, the percentage is increased from 0.0% to 90.0%.

We also analyzed the effects of the valid inequalities individually. Even though indi-

vidual effects are dependent on the previous set of constraints, we take the order of our finalized model into account and solved the same problems by adding the valid inequalities one by one into the model and produced the same percentage tables after each of those additions. The tables can be found in Appendix B. We provide the average percentages of the integral solutions from these tables for both y and x and for y only in Figure 7.1. In this figure, "Base" is used to represent the percentages of the solutions with the base model without any addition of valid inequalities.

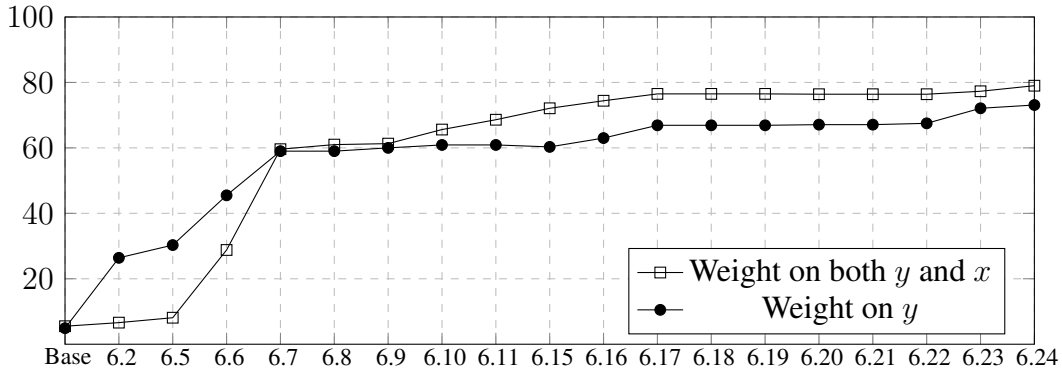


Figure 7.1: Average percentages of the integral solutions of trees with the addition of each valid inequality in the order of the finalized model.

In the figure, it can be seen that constraints 6.7, 6.6 and 6.2 create the most significant increases in average percentages, 22.13%, 17.97% and 11.27%, respectively. Fourth best increase is for Constraint 6.17 with 3.01%.

It is usually expected that an addition of valid inequality increases the percentage. However, there are some cases where the percentage decreases. For example, the average percentage of the problems with weight on only y decreases from 60.9% to 60.3% with the addition of Constraint 6.15. This is because in some problems, the model finds an integral solution even though there are other feasible non-integral solutions with the same objective value. By the addition of a new valid inequality, since the constraints are changed, the model might return one of the alternative non-integral solutions if the addition of new constraint does not cut the alternative solution, causing a decrease in the percentage. However, the same does not apply for the optimal value; the optimal objective function value of a problem cannot decrease (for a minimization problem) with an addition of a valid inequality. Thus, it might also be

helpful to find the "optimal objective function values" of the solutions returned.

Secondly, there might be a valid inequality where its improvement might be relatively small in terms of the percentage but relatively high in the optimal objective function value. Assume there exists a valid inequality, where change within the optimal objective function value of the LP relaxation model before and after the constraint is added is significant, but the solution is still returned non-integral after the addition. In the percentage study, it is shown as there is no change with the addition of this inequality, since both solutions are non-integral. However, we might have an objective function value which is much more closer (or even equal) to the optimal objective function value of the IP after the addition of the inequality. To see such effects, we did the same study for the optimal objective function values of the problems with the addition of each constraint, which can be found in Figure 7.2. Note that we did not split the solutions into two groups according their weights, similar to Figure 7.1, for this figure.

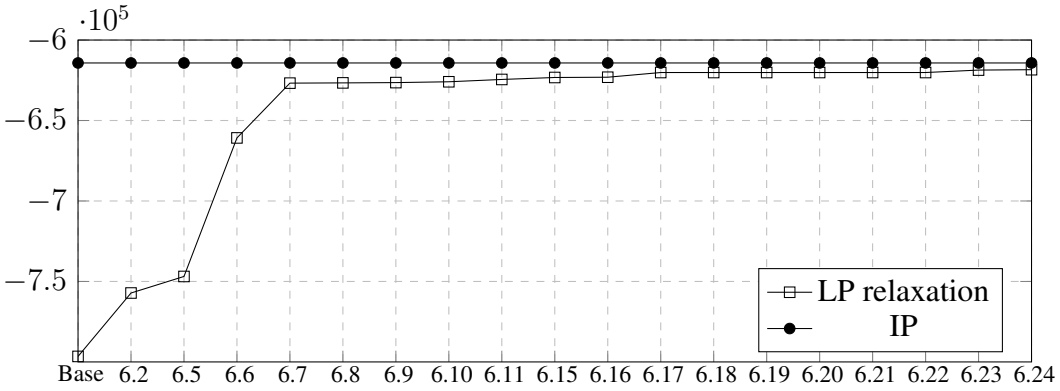


Figure 7.2: Total optimal objective function values of the solutions of trees with the addition of each valid inequality in the order of the finalized model.

The total optimal objective function values of the IP solutions are also provided, where the decision variables are restricted to be binaries. We observe that 97.70% of the difference of the total optimal objective function values in between the LP relaxation model of the base formulation to IP is cut by the addition of valid inequalities given in the finalized model. Additionally, to find the cumulative effects of the inequalities that are given after the finalized model in Proposition 6.23 to 6.25, we solved the LP relaxation by adding these inequalities and found that the percentage

increases from 97.70% to 98.06%.

We also observe that, similar to the percentage study, constraints 6.7, 6.6 and 6.2 increase the total optimal objective function values the most. However, the increase of the optimal objective function values are more significant for Constraint 6.6 then Constraint 6.7, even though Constraint 6.7 has the highest increase in the percentage of the integral solutions. Lastly, Constraint 6.5 also increases the objective significantly.

While these two figures show the solution quality, we would also like to analyze the solution times of these instances. For this, Figure 7.3 is provided.

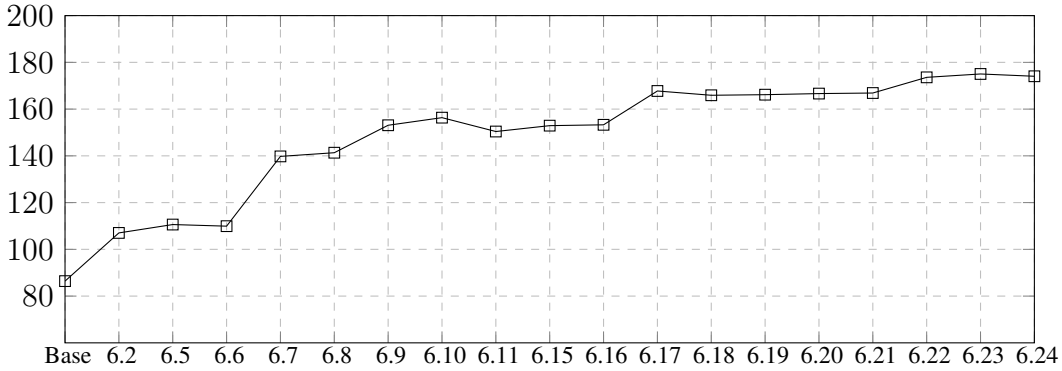


Figure 7.3: Total time of the solutions (in seconds) of trees with the addition of each valid inequality in the order of the finalized model.

We observe that Constraint 6.6 is very useful. With the addition of it, the percentage of the integral solutions increases by 17.97%, the average optimal objective function values increases by 47.21% of the difference between base LP relaxation to IP, and the solution time decreases by 0.62% compared to the case before its addition.

It is also realized that, Constraint 6.7 and 6.2, the remaining two of the top three constraints with the highest effects on both percentage and total optimal objective function value study, have the highest increase in solution time. Thus, there is a chance that their addition on an IP problem might increase the solution time. Even though their addition creates a stronger model which may decrease the solution time, the time of one LP relaxation solution with these constraints increases, for example, on a branch-and-bound algorithm for IP.

We observe that the constraints that are added earlier have generally better effects on

the solution quality and time. As mentioned earlier, their effects are dependent on the previous set of constraints. Observe that with the addition of the inequalities given after the finalized model, we found that the model decreases the difference between base LP relaxation and IP in terms of the optimal objective function value by 98.06%. This means that an additional of a valid inequality to our finalized model with the inequalities in Proposition 6.23 to 6.25 can only increase this percentage by 1.94% even though it might be very crucial, possibly defining the perfect neighborhood set polytope in trees. Thus, such a valid inequality has no chance to be seen as more effective then, for example, Constraint 6.6. Therefore, we wanted to do the same study but with a different order, and we take the opposite order of the finalize model. The same figures for the opposite order can be found in Figures 7.4, 7.5 and 7.6.

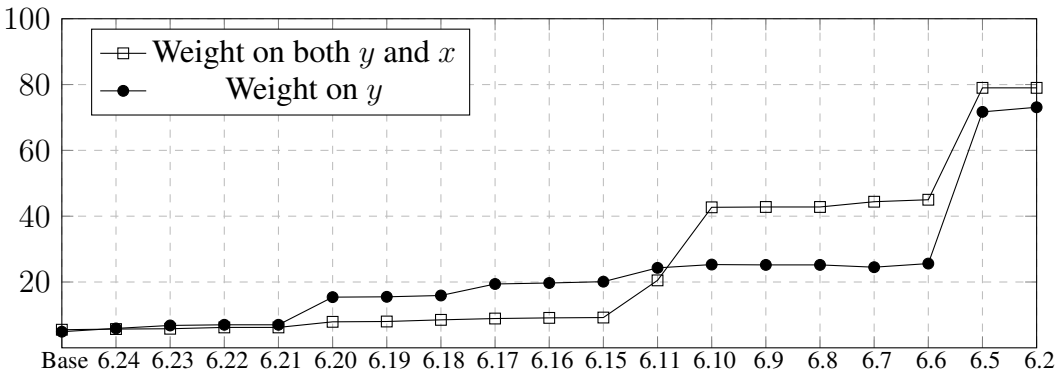


Figure 7.4: Average percentages of the integral solutions of trees with the addition of each valid inequality in the opposite order of the finalized model.

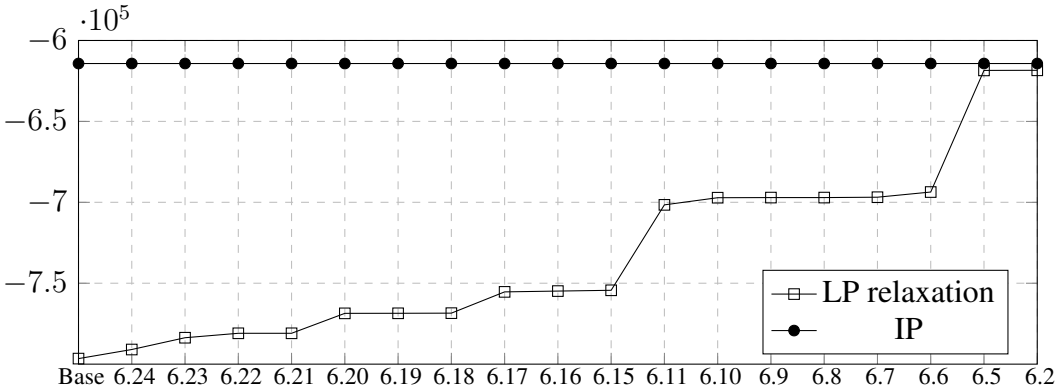


Figure 7.5: Total optimal objective function values of the solutions of trees with the addition of each valid inequality in the opposite order of the finalized model.

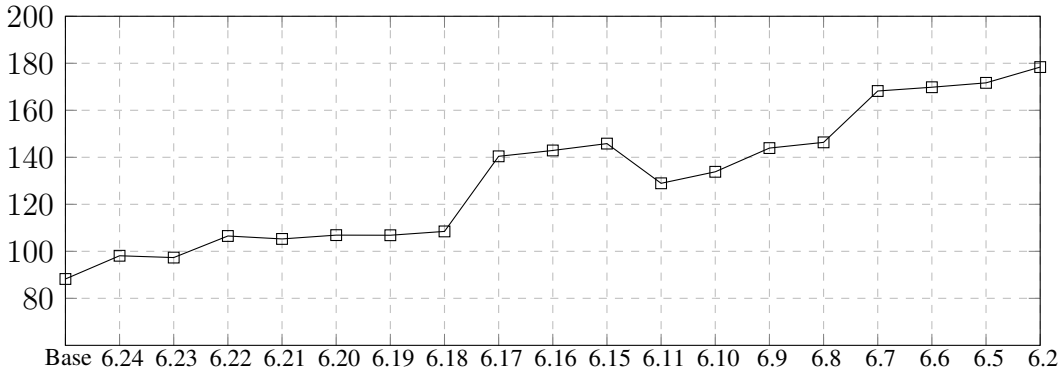


Figure 7.6: Total time of the solutions (in seconds) of trees with the addition of each valid inequality in the opposite order of the finalized model.

Lastly, we consider one random order of our inequalities and done the same study for the order. The figures for this order can be found in Figures 7.7, 7.8 and 7.9.

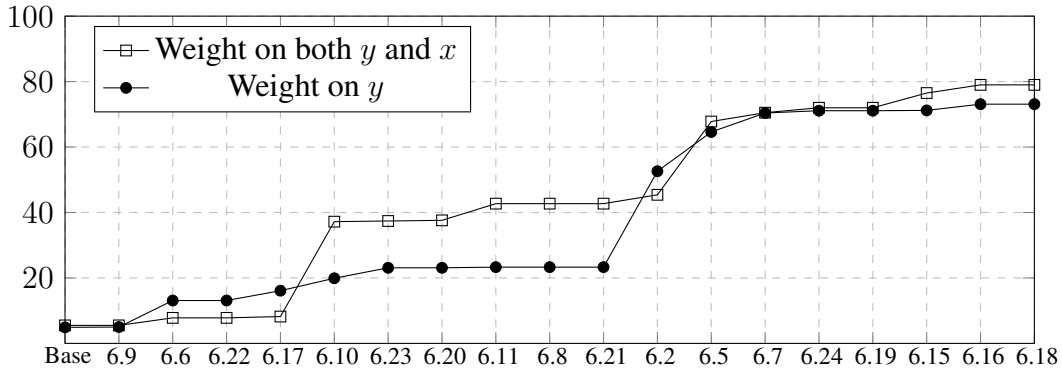


Figure 7.7: Average percentages of the integral solutions of trees with the addition of each valid inequality in the random order of the finalized model.

Combining the results for the three orders, we observe that constraints 6.5, 6.6, 6.11 and 6.23 are the most important valid inequalities for trees. Either the ratio of the increase in percentage and the optimal objective function values to the increase in solution time is the highest for them, or they are the ones that increase the percentage and the optimal objective function values while decreasing the solution time.

A second group of constraints are selected as 6.2, 6.10 and 6.24, which gives a good results in terms of the solution quality and time but not as good as the constraints that listed above for trees. We have made additional computational studies to find their

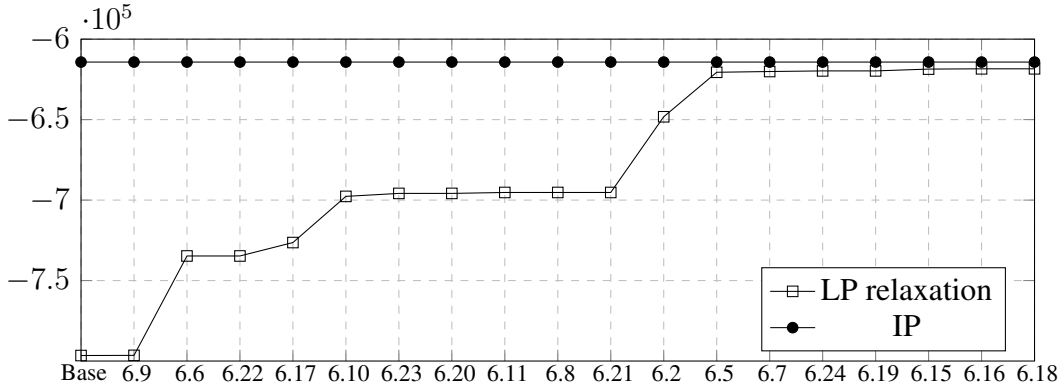


Figure 7.8: Total optimal objective function values of the solutions of trees with the addition of each valid inequality in the random order of the finalized model.

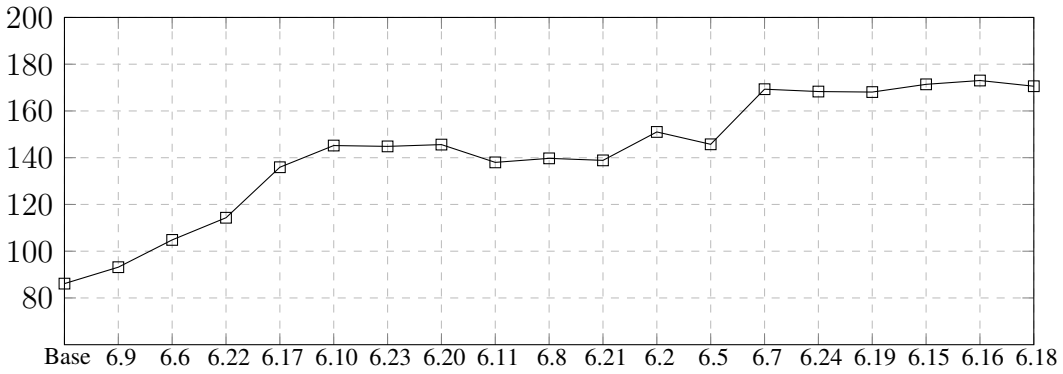


Figure 7.9: Total time of the solutions (in seconds) of trees with the addition of each valid inequality in the random order of the finalized model.

effects further.

Constraint 6.7 is not listed here because while it creates one of the most increases in the solution quality, it also increases solution time sharply, and the ratio to increase in quality to time is much smaller than those listed above.

7.2 Effects of the Valid Inequalities in $G(n, 2/n + 0.03)$

We made the same study that is done for the trees for arbitrary graphs in order to further study the constraints. To randomize an arbitrary graph, we used Erdős–Rényi random graph model with $G(n, p)$. Given two inputs n and p where n is a positive

integer and $0 \leq p \leq 1$, a graph with n vertices is randomized in which the probability of occurrence of each edge is p. We assumed that the graphs are connected (otherwise, each connected component can be input to the model individually), and randomized another graph if the output of $G(n, p)$ is not a connected graph.

We first considered graphs with low densities, where the set of number of vertices $|V| \in \{5, 10, 25, 50\}$ and $p = 2/n + 0.03$ are used. For each combination of $|V|$ and weight, we have randomized 500 connected graphs and solved them with LP relaxation to see the percentages of the integral solutions. The percentages for the base model without any valid inequalities, and the final model with all valid inequalities can be found in Table 7.3 and 7.4.

Table 7.3: Percentage of the integral solutions of 500 connected graphs of $G(n, 2/n + 0.03)$ with the base LP relaxation model.

$\|V\|$	weight on both y and x						weight on y					
	1	-1	$[0,1]$	$[-1,0]$	$[-1,1]$	Avg.	1	-1	$[0,1]$	$[-1,0]$	$[-1,1]$	Avg.
5	**0.0%**	2.0%	3.8%	18.6%	24.0%	9.7%	2.0%	1.8%	33.8%	4.0%	35.2%	15.4%
10	0.6%	**0.0%**	3.0%	5.6%	9.2%	3.7%	11.6%	**0.0%**	20.8%	**0.0%**	19.2%	10.3%
25	**0.0%**	**0.0%**	**0.0%**	**0.0%**	0.4%	0.1%	0.8%	**0.0%**	3.6%	**0.0%**	2.4%	1.4%
50	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	1.0%	**0.0%**	**0.0%**	0.2%
Avg.	0.2%	0.5%	1.7%	6.1%	8.4%	3.4%	3.6%	0.5%	14.8%	1.0%	14.2%	6.8%

Table 7.4: Percentage of the integral solutions of 500 connected graphs of $G(n, 2/n + 0.03)$ with the finalized model.

$\|V\|$	weight on both y and x						weight on y					
	1	-1	$[0,1]$	$[-1,0]$	$[-1,1]$	Avg.	1	-1	$[0,1]$	$[-1,0]$	$[-1,1]$	Avg.
5	52.0%	57.6%	58.8%	73.6%	81.4%	64.7%	60.2%	76.0%	**100.0%**	89.6%	95.8%	84.3%
10	24.4%	33.0%	30.6%	57.6%	43.4%	37.8%	52.6%	24.4%	91.8%	35.8%	62.6%	53.4%
25	3.2%	7.8%	3.8%	23.2%	9.6%	9.5%	55.8%	0.4%	73.4%	1.4%	17.8%	29.8%
50	**0.0%**	0.2%	**0.0%**	2.0%	**0.0%**	0.4%	11.0%	**0.0%**	45.8%	**0.0%**	0.2%	11.4%
Avg.	19.9%	24.7%	23.3%	39.1%	33.6%	28.1%	44.9%	25.2%	77.8%	31.7%	44.1%	44.7%

We observe that the effects of the valid inequalities decreases when we consider non-tree graphs, even though these graphs $G(n, 2/n + 0.03)$ are very close to trees. A graph with $G(n, 2/n)$ would have $n-1$ edges on average which is equal to the number of edges of a tree, and we only increase this probability by 0.03 in these instances. We

observe that the average percentages for both y and x, and for only y are increased from 3.4% and 6.8% to 28.1% and 44.7%, respectively.

Similar to the ones of trees, we have made the figures of these solutions in terms of the solution quality and time. These can be found in Figure 7.10, 7.11 and 7.12.

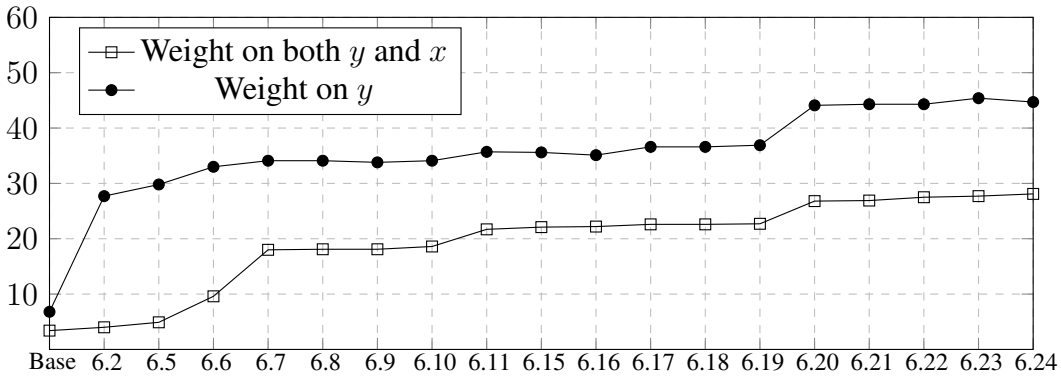


Figure 7.10: Average percentages of the integral solutions of $G(n, 2/n + 0.03)$ with the addition of each valid inequality in the order of the finalized model.

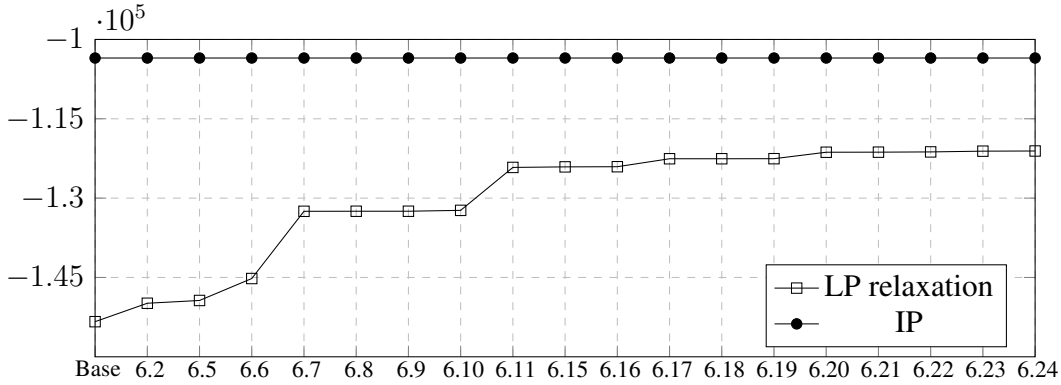


Figure 7.11: Total optimal objective function values of the solutions of $G(n, 2/n + 0.03)$ with the addition of each valid inequality in the order of the finalized model.

We observe that Constraint 6.20 creates a significant improvement on percentages, which is written for all maximal cliques C. In a tree, we have $|C| \leq 2$ which decreases the effect of the constraint. Additionally, other constraints that found to be helpful for trees are still the ones that increase the solution quality the most for the graphs $G(n, 2/n + 0.03)$.

Similar to the trees, Constraint 6.7 increases both solution quality and time. For

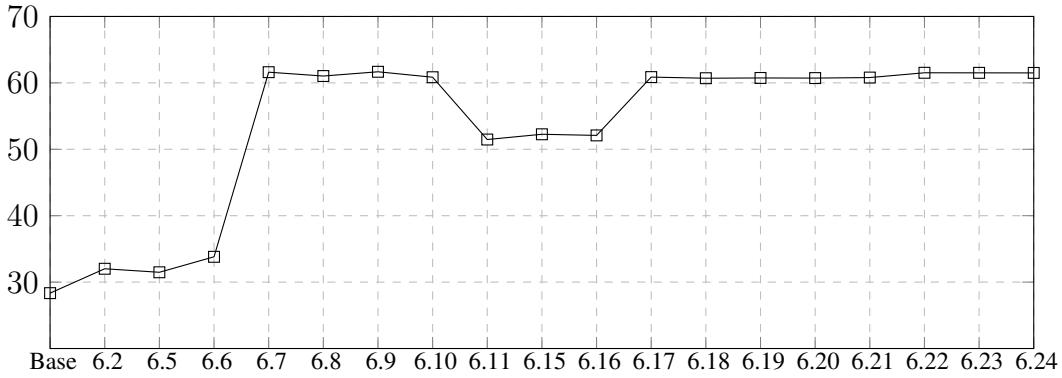


Figure 7.12: Total time of the solutions (in seconds) of $G(n, 2/n + 0.03)$ with the addition of each valid inequality in the order of the finalized model.

$G(n, 2/n + 0.03)$, time increases sharply with 82.22% with its addition, which decreases the ratio of the increase in solution quality to time from trees to $G(n, 2/n + 0.03)$.

7.3 Effects of the Valid Inequalities in $G(n, 0.8)$

Lastly, we have done the test for more densely connected graphs. We have created 200 graphs of $G(n, 0.8)$ for each weight and each number of vertices in $|V| \in \{5, 10, 25, 50\}$. The percentages for the base model without any valid inequalities, and the final model with all valid inequalities can be found in Table 7.5 and 7.6.

Table 7.5: Percentage of the integral solutions of 200 connected graphs of $G(n, 0.8)$ with the base LP relaxation model.

$\|V\|$	weight on both y and x						weight on y					
	1	-1	$[0,1]$	$[-1,0]$	$[-1,1]$	Avg.	1	-1	$[0,1]$	$[-1,0]$	$[-1,1]$	Avg.
5	**0.0%**	**0.0%**	**0.0%**	6.0%	14.0%	4.0%	**0.0%**	**0.0%**	7.0%	0.5%	27.5%	7.0%
10	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	4.5%	0.9%
25	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**
50	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**
Avg.	**0.0%**	**0.0%**	**0.0%**	1.5%	3.5%	1.0%	**0.0%**	**0.0%**	1.8%	0.1%	8.0%	2.0%

With the addition of valid inequalities, the model is capable of solving all the problems by the LP relaxation of the model for the weight on only y in the range $[0, 1]$,

Table 7.6: Percentage of the integral solutions of 200 connected graphs of $G(n, 0.8)$ with the finalized model.

$\|V\|$	weight on both y and x						weight on y					
	1	-1	$[0,1]$	$[-1,0]$	$[-1,1]$	Avg.	1	-1	$[0,1]$	$[-1,0]$	$[-1,1]$	Avg.
5	24.5%	41.5%	25.5%	73.0%	68.0%	46.5%	33.0%	82.0%	**100.0%**	83.5%	93.5%	78.4%
10	**0.0%**	**0.0%**	**0.0%**	4.5%	5.0%	1.9%	32.5%	1.5%	**100.0%**	3.0%	11.5%	29.7%
25	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**100.0%**	**0.0%**	**0.0%**	20.0%
50	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	0.5%	**0.0%**	**100.0%**	**0.0%**	**0.0%**	20.1%
Avg.	6.1%	10.4%	6.4%	19.4%	18.3%	12.1%	16.5%	20.9%	**100.0%**	21.6%	26.3%	37.1%

i.e., the weighted case of $\theta(G)$. We observed that this starts with the addition of Constraint 6.2, that means, the base model with the addition of Constraint 6.2 is capable of solving the weighted case of $\theta(G)$ for our instances, i.e., gives an average of 100.0% integral solutions with LP relaxation for these problems. This is one of the cuts that is found to be the most significant for the other test instances as well.

The figures for the solution quality and the time, similar to the previous instances, can be found in Figures 7.13, 7.14 and 7.15.

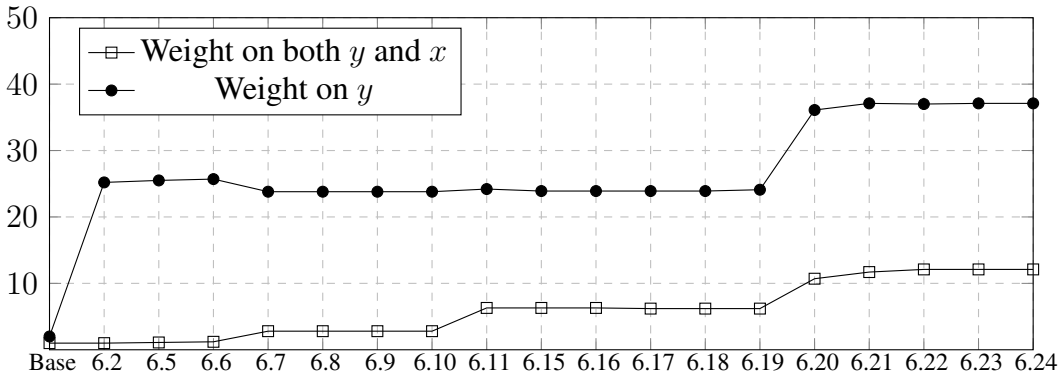


Figure 7.13: Average percentages of the integral solutions of $G(n, 0.8)$ with the addition of each valid inequality in the order of the finalized model.

In the solution quality, we observe that the same constraints that are found to be effective for the trees and the graphs of $G(n, 2/n + 0.03)$ are also effective for $G(n, 0.8)$. We observe almost no effect in terms of the percentage of the integral solutions and the increase in the total optimal objective function values for the constraints except 6.2, 6.7, 6.11 and 6.20 for $G(n, 0.8)$.

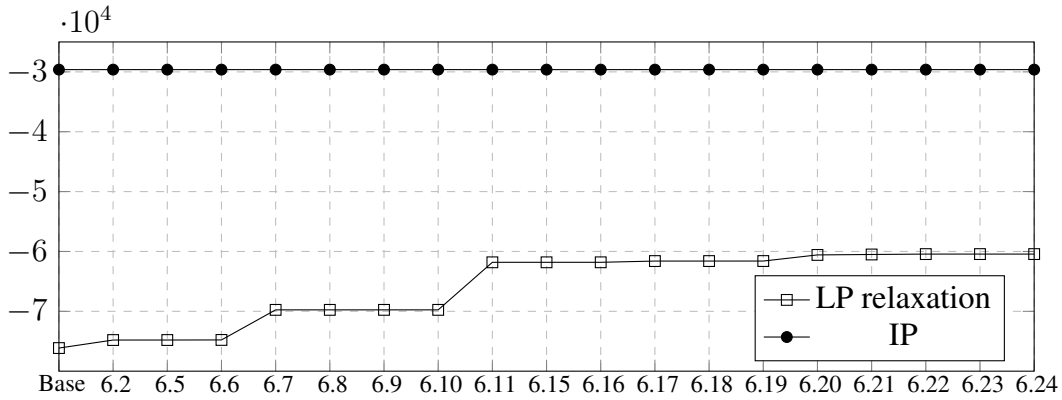


Figure 7.14: Total optimal objective function values of the solutions of $G(n, 0.8)$ with the addition of each valid inequality in the order of the finalized model.

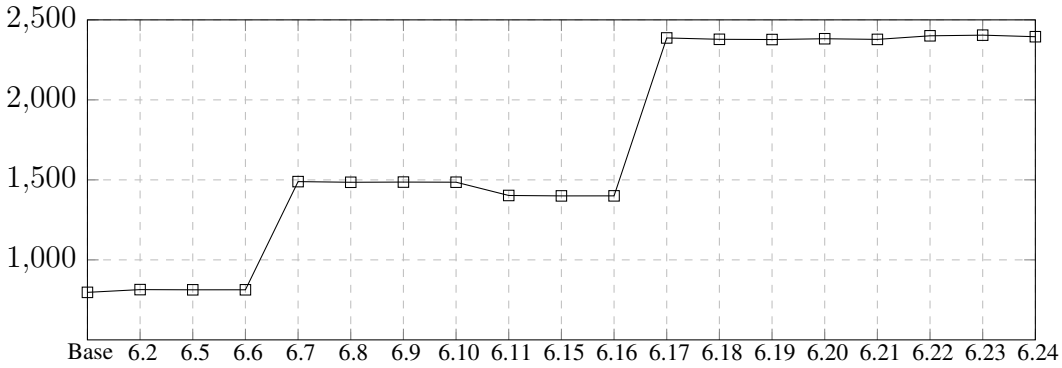


Figure 7.15: Total time of the solutions (in seconds) of $G(n, 0.8)$ with the addition of each valid inequality in the order of the finalized model.

Additionally, Constraint 6.7 again increases the time for these instances with Constraint 6.17. The increase in time for Constraint 6.17 occurs more drastically in $G(n, 0.8)$, then for $G(n, 2/n + 0.03)$. The reason behind this could be that the constraint is written for a vertex j which satisfies $|\Delta(j)| \geq 3$. In a more densely connected graph, this occurs more frequently, resulting increase in the number of constraints, and then possibly increase in time for one LP relaxation solution.

7.4 Selection of a Subset of Valid Inequalities

According to the results of our instances, we select a subset of additional valid inequalities in order to decrease the solution time of the IP model without any valid

inequality. We observed that Constraints 6.2, 6.7, 6.11 and 6.20 are much more effective in terms of the solution quality than the others, but Constraint 6.7 also increases the solution time significantly. The increases in percentage are 82.22% and 83.15% for the instances $G(n, 2/n + 0.03)$ and $G(n, 0.8)$, respectively. Similar to Constraint 6.7, we observe that Constraint 6.2 increases the solution time significantly as well. However, the increase is less than Constraint 6.7 and it is the one that increases the percentage of integral solutions with the weight on only y for graphs $G(n, 0.8)$, in the range $[0, 1]$, from 1.8% to 100.0%. Thus, we did not eliminate Constraint 6.2 and we have selected Constraints 6.2, 6.11 and 6.20 to be in our subset of valid inequalities by discarding only Constraint 6.7 from these constraints.

We have also made a ratio analysis and sorted the constraints in terms of their ratios of the increase in solution quality to increase in solution time (or possible decrease in solution time), which also enabled us to discard Constraint 6.7, for example. In this analysis, even though their effects are seen to be less significant, we observed that constraints 6.5, 6.6, 6.8, 6.16, 6.23, and 6.24 can be added into the subset.

In this list, constraints 6.5, 6.6, and 6.23 are mentioned to be effective on trees but their effects were smaller for arbitrary graphs. That is because all of these constraints are written more in loosely connected graphs, which maximizes their effects on trees. Yet, their ratios are one of the highest ones in our analysis for $G(n, 2/n + 0.03)$ and $G(n, 0.8)$ graphs as well.

Overall, we have created the subset of valid inequalities to be added into the model to possible decrease the solution time of IP. The subset is selected to be constraints 6.2, 6.5, 6.6, 6.8, 6.11, 6.16, 6.20, 6.23, and 6.24.

7.5 Effects of the Subset of the Valid Inequalities in the Solution Time of IP

We compare two models, the base IP model with the IP model with additional valid inequalities listed in the subset in terms of the solution time. We considered $G(n, p)$ model with the number of vertices values of $n = \{100, 150, 200\}$ and probability values of $p = \{3/n, 4/n, 5/n\}$. In all of the instances, we randomized the weights in the range of $[-1, 1]$ for both y and x. For each combination of n and p, we have

randomized 50 connected graphs. Moreover, we have selected a time limit for the solution of the model as 1 hour and used the default MIP gap limits of CPLEX. Average and median of the solution times for each combination can be found in Table 7.7 and 7.8, respectively. Additionally, ratios of the values of advanced model to base model are also provided. We call the model with additional valid inequalities advanced model.

Table 7.7: Average solution times (in seconds) of the 50 instances of each combination with the base model and the advanced model with the subset of valid inequalities.

	Base Model			Advanced Model			Ratio (Adv / Base)		
n \ p	$3/n$	$4/n$	$5/n$	$3/n$	$4/n$	$5/n$	$3/n$	$4/n$	$5/n$
100	1.00	4.13	10.03	0.53	2.45	6.94	0.54	0.59	0.69
150	7.19	38.11	153.5	3.90	30.31	118.6	0.54	0.80	0.77
200	33.02	476.9	2824	21.79	348.2	2723	0.66	0.73	0.96

Table 7.8: Median solution times (in seconds) of the 50 instances of each combination with the base model and the advanced model with the subset of valid inequalities.

	Base Model			Advanced Model			Ratio (Adv / Base)		
n \ p	$3/n$	$4/n$	$5/n$	$3/n$	$4/n$	$5/n$	$3/n$	$4/n$	$5/n$
100	0.96	2.61	10.09	0.41	1.71	4.79	0.43	0.65	0.47
150	4.35	26.06	72.76	2.11	25.48	60.61	0.48	0.98	0.83
200	23.95	128.1	3601	11.74	84.67	3709	0.49	0.66	1.03

Even though we found that the solution times for both models might be much more than 1 hour without a time limit in preliminary experiments, we observe that the MIP gap is found as negligibly small with a time limit of 1 hour in all instances. Average objective function values and best objective function values can be found in Table 7.9 and 7.10, respectively. "Diff." represents the difference.

We observe that 30% and 33% improvement occurred on the average and median of the solution times for our instances with the additional valid inequalities, respectively, while the MIP gap decreases with the additional valid inequalities.

Table 7.9: Average objective function value of the 50 instances of each combination with the base model and the advanced model with the subset of valid inequalities.

	Base Model			Advanced Model			Diff. (Adv - Base)		
n \ p	$3/n$	$4/n$	$5/n$	$3/n$	$4/n$	$5/n$	$3/n$	$4/n$	$5/n$
100	-30.07	-28.97	-29.75	-30.07	-28.97	-29.75	0.00	0.00	0.00
150	-43.90	-43.62	-44.03	-43.90	-43.62	-44.03	0.00	0.00	0.00
200	-58.98	-59.37	-58.86	-58.98	-59.38	-58.97	0.00	-0.01	-0.11

Table 7.10: Average best objective function value of the 50 instances of each combination with the base model and the advanced model with the subset of valid inequalities.

	Base Model			Advanced Model			Diff. (Adv - Base)		
n \ p	$3/n$	$4/n$	$5/n$	$3/n$	$4/n$	$5/n$	$3/n$	$4/n$	$5/n$
100	-30.07	-28.97	-29.75	-30.07	-28.97	-29.75	0.00	0.00	0.00
150	-43.91	-43.62	-44.03	-43.91	-43.62	-44.03	0.00	0.00	0.00
200	-58.99	-59.46	-60.27	-58.99	-59.44	-60.07	0.00	0.02	0.20

CHAPTER 8

CONCLUSION

In this thesis, we studied the minimum weighted perfect neighborhood set problem. Only, the unweighted version of this problem has been studied before in the literature. We first propose a linear time algorithm for trees for the MinWPNSP. This algorithm is a generalization of the algorithm proposed by Hedetniemi et al. (1997) [1]. Moreover, we have observed that the original algorithm had many flaws. We have corrected the multiplication table that is used in the original algorithm. To check the correctness of the generalized algorithm, we proposed an integer programming (IP) formulation for the MinWPNSP that is applicable to general graphs. We observed that the generalization of the algorithm of Hedetniemi et al. (1997) [1] runs much faster than the IP formulation on randomly generated trees.

We, then, made this formulation stronger with additional valid inequalities. We characterized the perfect neighborhood set polytope for star graphs and complete graphs by the help of the valid inequalities. Then, we made a detailed analysis of the individual effects of the valid inequalities, and selected a subset of these valid inequalities for the comparison. We compared the IP formulation with and without the selected subset of the valid inequalities in terms of the time and observed that the solution times decrease about 30% for the Erdős–Rényi random graph model with $G(n, p)$ with lower densities.

The MinWPNSP can be considered in future studies. In particular, specific algorithms can be designed for some graph classes that beat the IP formulation in terms of solution time. Moreover, the case when w_j's are all zero and v_j's are all 1 or -1 can deserve some attention.

REFERENCES

[1] S. Hedetniemi, S. Hedetniemi, and M. Henning, "The algorithmic complexity of perfect neighborhoods in graphs," *Journal of combinatorial mathematics and combinatorial computing*, vol. 25, pp. 183–192, 1997.

[2] G. H. Fricke, T. W. Haynes, S. Hedetniemi, S. T. Hedetniemi, and M. A. Henning, "On perfect neighborhood sets in graphs," *Discrete mathematics*, vol. 199, no. 1-3, pp. 221–225, 1999.

[3] E. J. Cockayne, S. M. Hedetniemi, S. T. Hedetniemi, and C. M. Mynhardt, "Irredundant and perfect neighbourhood sets in trees," *Discrete mathematics*, vol. 188, no. 1-3, pp. 253–260, 1998.

[4] E. Cockayne and C. Mynhardt, "On a conjecture concerning irredundant and perfect neighbourhood sets in graphs," *Journal of combinatorial mathematics and combinatorial computing*, vol. 31, pp. 241–253, 1999.

[5] O. Favaron and J. Puech, "Irredundant and perfect neighborhood sets in graphs and claw-free graphs," *Discrete mathematics*, vol. 197, pp. 269–284, 1999.

[6] T. Haynes, S. Hedetniemi, and P. Slater, "Domination in graphs: the theory," 1998.

[7] T. Xie, B. Zhong, and T. Peng, "On perfect neighborhood and irredundant sets in trees," *Journal of Mathematical Research with Applications*, vol. 27, pp. 13–18, 2007.

[8] O. Favaron, "From irredundance to annihilation: a brief overview of some domination parameters of graphs," 2000.

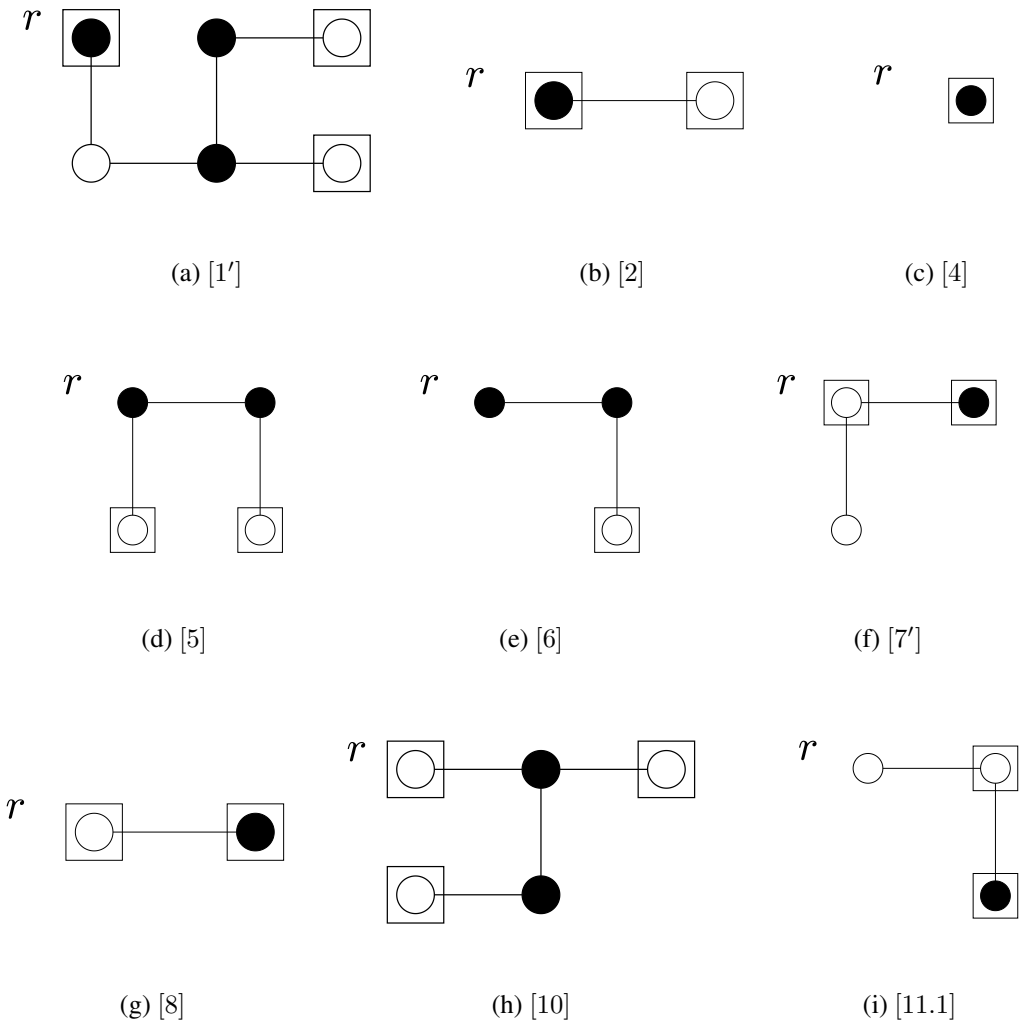
[9] S. T. Hedetniemi and T. Haynes, "Unsolved algorithmic problems on trees," *AKCE International Journal of Graphs and Combinatorics*, vol. 3, no. 1, pp. 1–37, 2006.

[10] J. Malık, “Problem 7. open perfect neighborhood sets on trees (suggested,” *KAMAK 2017 Penzion Orlice, Zámel September 17.–22.*, p. 9.

[11] E. Sampathkumar and P. Neeralagi, “Independent, perfect and connected neighbourhood numbers of a graph,” *J. Combin. Inf. Syst. Sci*, vol. 19, no. 3-4, pp. 139–145, 1994.

Appendix A

APPENDIX

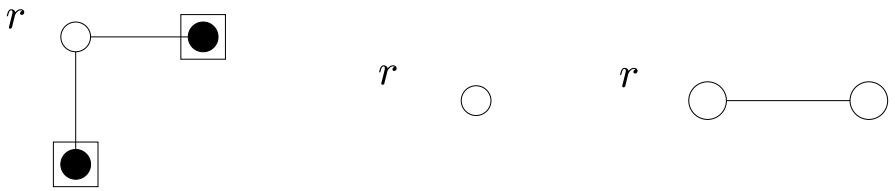


(a) $[1']$ (b) $[2]$ (c) $[4]$

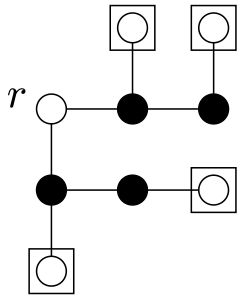
(d) $[5]$ (e) $[6]$ (f) $[7']$

(g) $[8]$ (h) $[10]$ (i) $[11.1]$

Figure A.1: Example of classes from $[1']$ to $[11.1]$.



(a) [11.2] (b) [12] (c) [13]



(d) [14]

Figure A.2: Example of classes from [11.2] to [14].

Appendix B

APPENDIX

Table B.1: Percentage of the integral solutions of 1000 trees after Constraint 6.2 is added to the base model.

$\|V\|$	weight on both y and x						weight on y					
	1	-1	$[0,1]$	$[-1,0]$	$[-1,1]$	Avg.	1	-1	$[0,1]$	$[-1,0]$	$[-1,1]$	Avg.
5	**0.0%**	9.3%	15.1%	41.0%	36.8%	20.4%	92.6%	9.3%	**100.0%**	8.1%	51.8%	52.4%
10	23.7%	13.5%	16.2%	20.4%	15.4%	17.8%	70.5%	0.6%	93.0%	0.9%	24.9%	38.0%
25	2.1%	0.3%	1.0%	1.4%	1.3%	1.2%	55.2%	**0.0%**	84.4%	**0.0%**	3.8%	28.7%
50	0.1%	**0.0%**	**0.0%**	**0.0%**	**0.0%**	0.0%	20.6%	**0.0%**	68.7%	**0.0%**	0.3%	17.9%
75	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	3.4%	**0.0%**	54.9%	**0.0%**	**0.0%**	11.7%
100	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	4.1%	**0.0%**	45.5%	**0.0%**	**0.0%**	9.9%
Avg.	4.3%	3.9%	5.4%	10.5%	8.9%	6.6%	41.1%	1.7%	74.4%	1.5%	13.5%	26.4%

Table B.2: Percentage of the integral solutions of 1000 trees after Constraint 6.5 is added to the model of Table B.1.

$\|V\|$	weight on both y and x						weight on y					
	1	-1	$[0,1]$	$[-1,0]$	$[-1,1]$	Avg.	1	-1	$[0,1]$	$[-1,0]$	$[-1,1]$	Avg.
5	**0.0%**	9.3%	30.2%	41.0%	42.5%	24.6%	**100.0%**	9.3%	**100.0%**	8.1%	51.8%	53.8%
10	25.8%	13.5%	30.0%	20.4%	19.2%	21.8%	92.1%	0.6%	93.0%	0.9%	24.9%	42.3%
25	2.2%	0.3%	4.3%	1.4%	2.2%	2.1%	88.4%	**0.0%**	84.4%	**0.0%**	3.8%	35.3%
50	0.1%	**0.0%**	**0.0%**	**0.0%**	0.1%	0.0%	45.4%	**0.0%**	68.7%	**0.0%**	0.3%	22.9%
75	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	10.4%	**0.0%**	54.9%	**0.0%**	**0.0%**	13.1%
100	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	**0.0%**	25.0%	**0.0%**	45.5%	**0.0%**	**0.0%**	14.1%
Avg.	4.7%	3.9%	10.8%	10.5%	10.7%	8.1%	60.2%	1.7%	74.4%	1.5%	13.5%	30.3%

Table B.3: Percentage of the integral solutions of 1000 trees after Constraint 6.6 is added to the model of Table B.2.

$\|V\|$	weight on both y and x						weight on y					
	1	-1	$[0,1]$	$[-1,0]$	$[-1,1]$	Avg.	1	-1	$[0,1]$	$[-1,0]$	$[-1,1]$	Avg.
5	**100.0%**	9.3%	93.5%	47.3%	82.3%	66.5%	**100.0%**	66.1%	**100.0%**	74.4%	87.8%	85.7%
10	72.1%	14.5%	81.9%	25.3%	62.0%	51.2%	96.7%	34.9%	93.0%	48.1%	71.6%	68.9%
25	57.4%	0.3%	58.5%	3.0%	27.0%	29.2%	91.6%	5.8%	84.4%	15.5%	40.8%	47.6%
50	29.7%	**0.0%**	33.9%	0.1%	7.9%	14.3%	72.0%	0.7%	68.7%	2.7%	17.1%	32.2%
75	14.5%	**0.0%**	19.6%	**0.0%**	3.2%	7.5%	43.9%	**0.0%**	54.9%	0.2%	7.1%	21.2%
100	8.4%	**0.0%**	10.3%	**0.0%**	1.2%	4.0%	37.9%	**0.0%**	45.5%	**0.0%**	3.6%	17.4%
Avg.	47.0%	4.0%	49.6%	12.6%	30.6%	28.8%	73.7%	17.9%	74.4%	23.5%	38.0%	45.5%

Table B.4: Percentage of the integral solutions of 1000 trees after Constraint 6.7 is added to the model of Table B.3.

$\|V\|$	weight on both y and x						weight on y					
	1	-1	$[0,1]$	$[-1,0]$	$[-1,1]$	Avg.	1	-1	$[0,1]$	$[-1,0]$	$[-1,1]$	Avg.
5	**100.0%**	**100.0%**	93.5%	92.9%	89.0%	95.1%	**100.0%**	**100.0%**	**100.0%**	79.5%	90.0%	93.9%
10	73.1%	99.9%	83.5%	86.3%	72.6%	83.1%	99.8%	74.2%	96.8%	56.2%	77.8%	81.0%
25	57.4%	99.5%	64.2%	69.6%	46.4%	67.4%	99.4%	43.2%	95.2%	23.1%	51.7%	62.5%
50	30.9%	97.7%	40.3%	48.0%	22.5%	47.9%	91.3%	16.0%	90.1%	5.1%	27.4%	46.0%
75	15.4%	93.2%	24.2%	32.6%	10.4%	35.2%	81.7%	6.6%	85.0%	0.9%	13.9%	37.6%
100	9.7%	91.3%	13.8%	23.4%	5.8%	28.8%	75.8%	2.2%	79.2%	**0.0%**	7.0%	32.8%
Avg.	47.8%	96.9%	53.3%	58.8%	41.1%	59.6%	91.3%	40.4%	91.1%	27.5%	44.6%	59.0%

Table B.5: Percentage of the integral solutions of 1000 trees after Constraint 6.8 is added to the model of Table B.4.

$\|V\|$	weight on both y and x						weight on y					
	1	-1	$[0,1]$	$[-1,0]$	$[-1,1]$	Avg.	1	-1	$[0,1]$	$[-1,0]$	$[-1,1]$	Avg.
5	**100.0%**	**100.0%**	93.5%	95.3%	89.4%	95.6%	**100.0%**	**100.0%**	**100.0%**	79.5%	90.2%	93.9%
10	73.1%	99.8%	83.5%	89.0%	73.2%	83.7%	99.8%	74.2%	96.8%	56.2%	78.0%	81.0%
25	57.4%	99.5%	64.2%	75.6%	47.6%	68.9%	99.4%	43.2%	95.2%	23.1%	52.7%	62.7%
50	31.0%	97.7%	40.3%	57.8%	23.1%	50.0%	91.0%	16.4%	90.1%	5.1%	28.1%	46.1%
75	15.8%	94.3%	24.2%	41.0%	11.0%	37.3%	81.8%	6.4%	85.0%	0.9%	14.3%	37.7%
100	10.0%	90.9%	13.8%	31.7%	6.0%	30.5%	74.4%	2.1%	79.2%	**0.0%**	7.3%	32.6%
Avg.	47.9%	97.0%	53.3%	65.1%	41.7%	61.0%	91.1%	40.4%	91.1%	27.5%	45.1%	59.0%

Table B.6: Percentage of the integral solutions of 1000 trees after Constraint 6.9 is added to the model of Table B.5.

$\|V\|$	weight on both y and x						weight on y					
	1	-1	$[0,1]$	$[-1,0]$	$[-1,1]$	Avg.	1	-1	$[0,1]$	$[-1,0]$	$[-1,1]$	Avg.
5	**100.0%**	**100.0%**	94.2%	95.3%	90.1%	95.9%	**100.0%**	**100.0%**	**100.0%**	79.5%	91.8%	94.3%
10	72.7%	99.8%	83.5%	89.0%	74.5%	83.9%	**100.0%**	74.2%	98.2%	56.2%	79.8%	81.7%
25	57.5%	99.5%	64.3%	75.6%	49.3%	69.2%	99.7%	43.2%	96.6%	23.1%	54.1%	63.3%
50	31.0%	97.3%	40.7%	57.8%	25.8%	50.5%	92.2%	16.4%	93.8%	5.1%	29.8%	47.5%
75	16.0%	94.5%	24.5%	41.0%	12.0%	37.6%	82.0%	6.8%	89.7%	0.9%	15.7%	39.0%
100	9.1%	92.2%	14.2%	31.7%	6.6%	30.8%	76.0%	2.1%	85.3%	**0.0%**	8.3%	34.3%
Avg.	47.7%	97.2%	53.6%	65.1%	43.1%	61.3%	91.7%	40.5%	93.9%	27.5%	46.6%	60.0%

Table B.7: Percentage of the integral solutions of 1000 trees after Constraint 6.10 is added to the model of Table B.6.

$\|V\|$	weight on both y and x						weight on y					
	1	-1	$[0,1]$	$[-1,0]$	$[-1,1]$	Avg.	1	-1	$[0,1]$	$[-1,0]$	$[-1,1]$	Avg.
5	**100.0%**	**100.0%**	94.2%	98.0%	91.9%	96.8%	**100.0%**	**100.0%**	**100.0%**	80.2%	93.0%	94.6%
10	72.8%	99.8%	83.5%	95.5%	79.0%	86.1%	**100.0%**	74.2%	98.2%	57.1%	82.5%	82.4%
25	57.5%	99.7%	64.3%	88.0%	56.8%	73.3%	99.7%	43.4%	96.6%	24.9%	59.1%	64.7%
50	30.0%	97.9%	40.7%	78.8%	33.5%	56.2%	91.2%	16.8%	93.8%	5.7%	35.1%	48.5%
75	16.4%	96.7%	24.5%	66.6%	17.1%	44.3%	80.9%	6.4%	89.7%	0.9%	19.6%	39.5%
100	9.8%	94.2%	14.2%	58.2%	9.5%	37.2%	78.7%	2.2%	85.3%	0.1%	11.4%	35.5%
Avg.	47.8%	98.1%	53.6%	80.9%	48.0%	65.6%	91.8%	40.5%	93.9%	28.2%	50.1%	60.9%

Table B.8: Percentage of the integral solutions of 1000 trees after Constraint 6.11 is added to the model of Table B.7.

$\|V\|$	weight on both y and x						weight on y					
	1	-1	$[0,1]$	$[-1,0]$	$[-1,1]$	Avg.	1	-1	$[0,1]$	$[-1,0]$	$[-1,1]$	Avg.
5	**100.0%**	**100.0%**	94.2%	99.5%	92.9%	97.3%	**100.0%**	**100.0%**	**100.0%**	82.6%	94.5%	95.4%
10	73.0%	**100.0%**	84.6%	98.0%	82.0%	87.5%	94.1%	76.8%	98.2%	61.9%	84.7%	83.1%
25	57.7%	99.9%	65.5%	93.6%	61.7%	75.7%	97.9%	45.3%	96.6%	28.8%	63.6%	66.4%
50	31.3%	99.7%	41.2%	88.2%	39.3%	59.9%	91.8%	18.1%	94.6%	8.6%	40.7%	50.8%
75	16.2%	99.1%	25.7%	80.4%	21.1%	48.5%	62.6%	8.3%	90.7%	1.7%	24.3%	37.5%
100	10.0%	99.0%	15.7%	73.4%	13.7%	42.4%	55.3%	3.3%	87.0%	0.3%	15.7%	32.3%
Avg.	48.0%	99.6%	54.5%	88.9%	51.8%	68.6%	83.6%	42.0%	94.5%	30.7%	53.9%	60.9%

Table B.9: Percentage of the integral solutions of 1000 trees after Constraint 6.15 is added to the model of Table B.8.

$\|V\|$	weight on both y and x						weight on y					
	1	-1	$[0,1]$	$[-1,0]$	$[-1,1]$	Avg.	1	-1	$[0,1]$	$[-1,0]$	$[-1,1]$	Avg.
5	**100.0%**	**100.0%**	96.5%	99.5%	93.6%	97.9%	**100.0%**	**100.0%**	**100.0%**	82.6%	94.5%	95.4%
10	83.0%	**100.0%**	87.7%	98.0%	83.4%	90.4%	95.0%	77.0%	98.2%	61.9%	84.7%	83.4%
25	70.1%	99.9%	69.7%	93.6%	63.4%	79.3%	97.0%	45.4%	96.6%	28.8%	63.6%	66.3%
50	46.9%	99.7%	47.2%	88.2%	40.7%	64.5%	76.2%	18.1%	94.6%	8.6%	40.7%	47.6%
75	32.0%	99.4%	32.6%	80.4%	23.0%	53.5%	60.4%	8.2%	90.7%	1.7%	24.3%	37.1%
100	24.3%	98.8%	22.6%	73.4%	15.4%	46.9%	55.1%	3.2%	87.0%	0.3%	15.7%	32.3%
Avg.	59.4%	99.6%	59.4%	88.9%	53.3%	72.1%	80.6%	42.0%	94.5%	30.7%	53.9%	60.3%

Table B.10: Percentage of the integral solutions of 1000 trees after Constraint 6.16 is added to the model of Table B.9.

$\|V\|$	weight on both y and x						weight on y					
	1	-1	$[0,1]$	$[-1,0]$	$[-1,1]$	Avg.	1	-1	$[0,1]$	$[-1,0]$	$[-1,1]$	Avg.
5	**100.0%**	**100.0%**	97.1%	99.5%	94.3%	98.2%	**100.0%**	**100.0%**	**100.0%**	82.6%	94.5%	95.4%
10	87.3%	**100.0%**	90.6%	98.0%	84.1%	92.0%	98.1%	76.9%	98.2%	61.9%	84.7%	84.0%
25	75.0%	99.9%	73.6%	93.6%	65.7%	81.6%	98.6%	45.4%	96.6%	28.8%	63.6%	66.6%
50	54.5%	99.6%	53.8%	88.2%	43.6%	67.9%	93.9%	18.2%	94.6%	8.6%	40.7%	51.2%
75	40.1%	99.2%	37.8%	80.4%	25.4%	56.6%	87.4%	8.2%	90.7%	1.7%	24.3%	42.5%
100	31.8%	98.8%	28.3%	73.4%	17.4%	49.9%	86.1%	3.2%	87.0%	0.3%	15.7%	38.5%
Avg.	64.8%	99.6%	63.5%	88.9%	55.1%	74.4%	94.0%	42.0%	94.5%	30.7%	53.9%	63.0%

Table B.11: Percentage of the integral solutions of 1000 trees after Constraint 6.17 is added to the model of Table B.10.

$\|V\|$	weight on both y and x						weight on y					
	1	-1	$[0,1]$	$[-1,0]$	$[-1,1]$	Avg.	1	-1	$[0,1]$	$[-1,0]$	$[-1,1]$	Avg.
5	**100.0%**	**100.0%**	98.2%	99.5%	95.3%	98.6%	**100.0%**	**100.0%**	**100.0%**	86.8%	96.2%	96.6%
10	87.2%	**100.0%**	91.2%	98.7%	86.2%	92.7%	98.3%	80.6%	98.2%	75.3%	88.9%	88.3%
25	75.9%	99.9%	75.0%	96.4%	69.3%	83.3%	98.1%	52.3%	96.6%	46.0%	70.6%	72.7%
50	55.6%	99.8%	56.5%	93.6%	48.3%	70.8%	93.6%	25.4%	94.6%	19.8%	50.1%	56.7%
75	41.3%	99.3%	41.2%	87.2%	29.4%	59.7%	86.1%	13.6%	90.7%	8.8%	32.9%	46.4%
100	34.8%	99.0%	32.1%	82.8%	20.5%	53.8%	84.3%	5.4%	87.0%	3.0%	24.7%	40.9%
Avg.	65.8%	99.7%	65.7%	93.0%	58.2%	76.5%	93.4%	46.2%	94.5%	40.0%	60.6%	66.9%

Table B.12: Percentage of the integral solutions of 1000 trees after Constraint 6.18 is added to the model of Table B.11.

$\|V\|$	weight on both y and x						weight on y					
	1	-1	$[0,1]$	$[-1,0]$	$[-1,1]$	Avg.	1	-1	$[0,1]$	$[-1,0]$	$[-1,1]$	Avg.
5	**100.0%**	**100.0%**	98.2%	99.5%	96.1%	98.8%	**100.0%**	**100.0%**	**100.0%**	86.8%	96.2%	96.6%
10	87.2%	**100.0%**	91.2%	98.7%	86.2%	92.7%	98.3%	80.6%	98.2%	75.3%	88.9%	88.3%
25	75.9%	99.9%	75.0%	96.4%	69.3%	83.3%	98.1%	52.3%	96.6%	46.0%	70.6%	72.7%
50	55.6%	99.8%	56.5%	93.6%	48.3%	70.8%	93.6%	25.4%	94.6%	19.8%	50.1%	56.7%
75	41.3%	99.3%	41.2%	87.2%	29.4%	59.7%	86.1%	13.6%	90.7%	8.8%	32.9%	46.4%
100	34.8%	99.0%	32.1%	82.8%	20.5%	53.8%	84.3%	5.4%	87.0%	3.0%	24.7%	40.9%
Avg.	65.8%	99.7%	65.7%	93.0%	58.3%	76.5%	93.4%	46.2%	94.5%	40.0%	60.6%	66.9%

Table B.13: Percentage of the integral solutions of 1000 trees after Constraint 6.19 is added to the model of Table B.12.

$\|V\|$	weight on both y and x						weight on y					
	1	-1	$[0,1]$	$[-1,0]$	$[-1,1]$	Avg.	1	-1	$[0,1]$	$[-1,0]$	$[-1,1]$	Avg.
5	**100.0%**	**100.0%**	98.2%	99.5%	96.1%	98.8%	**100.0%**	**100.0%**	**100.0%**	86.8%	96.2%	96.6%
10	87.2%	**100.0%**	91.2%	98.7%	86.2%	92.7%	98.3%	80.6%	98.2%	75.3%	88.9%	88.3%
25	75.9%	99.9%	75.0%	96.4%	69.3%	83.3%	98.1%	52.3%	96.6%	46.0%	70.6%	72.7%
50	55.6%	99.8%	56.5%	93.6%	48.3%	70.8%	93.6%	25.4%	94.6%	19.8%	50.1%	56.7%
75	41.3%	99.3%	41.2%	87.2%	29.4%	59.7%	86.1%	13.6%	90.7%	8.8%	32.9%	46.4%
100	34.8%	99.0%	32.1%	82.8%	20.5%	53.8%	84.3%	5.4%	87.0%	3.0%	24.7%	40.9%
Avg.	65.8%	99.7%	65.7%	93.0%	58.3%	76.5%	93.4%	46.2%	94.5%	40.0%	60.6%	66.9%

Table B.14: Percentage of the integral solutions of 1000 trees after Constraint 6.20 is added to the model of Table B.13.

$\|V\|$	weight on both y and x						weight on y					
	1	-1	$[0,1]$	$[-1,0]$	$[-1,1]$	Avg.	1	-1	$[0,1]$	$[-1,0]$	$[-1,1]$	Avg.
5	**100.0%**	**100.0%**	98.2%	99.5%	96.1%	98.8%	**100.0%**	**100.0%**	**100.0%**	86.8%	96.2%	96.6%
10	87.0%	**100.0%**	91.2%	98.7%	86.2%	92.6%	99.5%	80.4%	98.2%	75.3%	88.9%	88.5%
25	75.9%	99.9%	75.0%	96.4%	69.3%	83.3%	99.1%	51.7%	96.6%	46.0%	70.6%	72.8%
50	55.6%	99.8%	56.5%	93.6%	48.3%	70.8%	93.5%	26.2%	94.6%	19.8%	50.1%	56.8%
75	39.8%	99.2%	41.2%	87.2%	29.4%	59.4%	87.7%	14.2%	90.7%	8.8%	32.9%	46.9%
100	34.2%	99.0%	32.1%	82.8%	20.5%	53.7%	84.4%	5.3%	87.0%	3.0%	24.7%	40.9%
Avg.	65.4%	99.7%	65.7%	93.0%	58.3%	76.4%	94.0%	46.3%	94.5%	40.0%	60.6%	67.1%

Table B.15: Percentage of the integral solutions of 1000 trees after Constraint 6.21 is added to the model of Table B.14.

$\|V\|$	weight on both y and x						weight on y					
	1	-1	$[0,1]$	$[-1,0]$	$[-1,1]$	Avg.	1	-1	$[0,1]$	$[-1,0]$	$[-1,1]$	Avg.
5	**100.0%**	**100.0%**	98.2%	99.5%	96.1%	98.8%	**100.0%**	**100.0%**	**100.0%**	86.8%	96.2%	96.6%
10	87.0%	**100.0%**	91.2%	98.7%	86.2%	92.6%	99.5%	80.4%	98.2%	75.3%	88.9%	88.5%
25	75.9%	99.9%	75.0%	96.4%	69.3%	83.3%	99.1%	51.7%	96.6%	46.0%	70.6%	72.8%
50	55.6%	99.8%	56.5%	93.6%	48.3%	70.8%	93.5%	26.2%	94.6%	19.8%	50.1%	56.8%
75	39.8%	99.2%	41.2%	87.2%	29.4%	59.4%	87.7%	14.2%	90.7%	8.8%	32.9%	46.9%
100	34.2%	99.0%	32.1%	82.8%	20.5%	53.7%	84.4%	5.3%	87.0%	3.0%	24.7%	40.9%
Avg.	65.4%	99.7%	65.7%	93.0%	58.3%	76.4%	94.0%	46.3%	94.5%	40.0%	60.6%	67.1%

Table B.16: Percentage of the integral solutions of 1000 trees after Constraint 6.22 is added to the model of Table B.15.

$\|V\|$	weight on both y and x						weight on y					
	1	-1	$[0,1]$	$[-1,0]$	$[-1,1]$	Avg.	1	-1	$[0,1]$	$[-1,0]$	$[-1,1]$	Avg.
5	**100.0%**	**100.0%**	98.2%	99.5%	96.1%	98.8%	**100.0%**	**100.0%**	**100.0%**	86.8%	96.2%	96.6%
10	87.0%	**100.0%**	91.2%	98.7%	86.2%	92.6%	**100.0%**	80.4%	98.2%	75.3%	88.9%	88.6%
25	75.7%	99.9%	75.0%	96.4%	69.3%	83.3%	99.4%	52.2%	96.6%	46.0%	70.6%	73.0%
50	55.2%	99.7%	56.5%	93.6%	48.3%	70.7%	93.1%	26.3%	94.6%	19.8%	50.1%	56.8%
75	40.3%	99.3%	41.2%	87.2%	29.4%	59.5%	93.8%	15.4%	90.7%	8.8%	32.9%	48.3%
100	34.3%	98.9%	32.1%	82.8%	20.5%	53.7%	89.6%	5.5%	87.1%	3.0%	24.7%	42.0%
Avg.	65.4%	99.6%	65.7%	93.0%	58.3%	76.4%	96.0%	46.6%	94.5%	40.0%	60.6%	67.5%

Table B.17: Percentage of the integral solutions of 1000 trees after Constraint 6.23 is added to the model of Table B.16.

$\|V\|$	weight on both y and x						weight on y					
	1	-1	$[0,1]$	$[-1,0]$	$[-1,1]$	Avg.	1	-1	$[0,1]$	$[-1,0]$	$[-1,1]$	Avg.
5	**100.0%**	**100.0%**	98.3%	**100.0%**	97.7%	99.2%	**100.0%**	**100.0%**	**100.0%**	**100.0%**	99.8%	100.0%
10	86.9%	**100.0%**	91.7%	98.9%	88.2%	93.1%	99.8%	91.9%	98.2%	81.9%	92.6%	92.9%
25	76.1%	99.9%	76.3%	96.8%	71.8%	84.2%	99.4%	66.3%	96.6%	55.2%	75.1%	78.5%
50	55.7%	99.8%	57.7%	94.4%	50.9%	71.7%	93.1%	38.3%	94.6%	28.2%	55.5%	61.9%
75	40.3%	99.3%	42.8%	88.5%	32.1%	60.6%	94.0%	25.9%	90.7%	14.9%	39.7%	53.0%
100	34.3%	99.1%	33.3%	83.7%	24.3%	54.9%	89.9%	15.6%	87.1%	7.2%	31.1%	46.2%
Avg.	65.6%	99.7%	66.7%	93.7%	60.8%	77.3%	96.0%	56.3%	94.5%	47.9%	65.6%	72.1%