SEQUENTIAL AND SIMULTANEOUS LIFTING IN THE NODE PACKING POLYHEDRON

by

JEFFREY WILLIAM PAVELKA

B.S., Kansas State University, 2011

A THESIS

Submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Industrial and Manufacturing Systems Engineering

College of Engineering

KANSAS STATE UNIVERSITY

Manhattan, Kansas

2011

Approved by:

Major Professor

Dr. Todd Easton

ABSTRACT

Integer programs (IPs) are a commonly researched class of decision problems. These problems are used in various applications to help companies, governments, or individuals make better decisions by determining optimal resource allocations. While IPs are practical tools, they require an exponential amount of effort to solve, unless $\mathcal{P} = \mathcal{NP}$. This fact has led to much research focused on reducing the time required to solve IPs.

Cutting planes are a commonly used tool for reducing IP solving time. Lifting, a process of changing the coefficients in an inequality, is often employed to strengthen cutting planes. When lifting, the goal is often to create a facet defining inequality, which is theoretically the strongest cutting plane.

This thesis introduces two new lifting procedures for the Node Packing problem. The Node Packing problem seeks to select the maximum number of nodes in a graph such that no two nodes are adjacent. The first lifting method, the Simultaneous Lifting Expansion, takes two inequalities and combines them to make a stronger cut. It works for any two general classes of inequalities, as long as the requisite graph structures are met.

The second method, the Cliques On Odd-holes Lifting (COOL) procedure, lifts from an odd-hole inequality to a facet defining inequality. COOL makes use of the Odd Gap Lifting procedure, an efficient method for finding lifting coefficients on odd holes. A computational study shows COOL to be effective in creating cuts in graphs with low edge densities.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In the field of mathematical optimization, Integer Programs (IPs) are a powerful tool. IPs are optimization problems where some or all of the decision variables must take on integer values. Restricting variables to integer values is often necessary in modeling decision problems, making IPs widely applicable for a vast array of industries. For example, truck routing [33, 46, 47, 1], sports scheduling [16, 48], and capital budgeting [19, 31] problems are all industry applications that have been modeled and improved using IPs.

Though they are useful in modeling various applications, IPs are difficult to solve. The most common method for solving IPs is called Branch and Bound. The first step in this method is to solve the IP's linear relaxation, which is identical to the IP except that no variable is required to be integer. This linear relaxation is said to be the root node of the branching tree. If the solution contains any non-integer variables, branching

occurs.

When branching from a node in the tree, two new problems are created. Each of these problems, called child problems or child nodes, include every constraint from the parent node. In addition, each child node contains a new constraint on one of the non-integer variables. For example, say that $x_1 = 2.4$ in the linear relaxation solution. Of the two child problems created, one of them includes the constraint $x \leq 2$. In the other, the constraint $x \geq 3$ is added. This procedure continues until all nodes are fathomed.

No branching will occur from a node that has been fathomed. Fathoming occurs under three conditions: When the problem is infeasible, when the linear relaxation solves to an integer value, or when the objective value for the node is worse than the best known integer solution.

Clearly, this method can be quite inefficient, since exponentially many branches may be created before a solution is found. Unfortunately, IPs are known to be $\mathcal{NP}$-Hard [32]. This means that there is no algorithm that can efficiently solve a general problem, unless $\mathcal{P} = \mathcal{NP}$. For this reason, there has been much research focused on developing methods to decrease IP's solving time.

One research area that can improve IP's solving time is cutting planes. Cutting planes are inequalities that eliminate a portion of an IP's linear relaxation space. They are said to be valid if and only if they are not violated by any integer solution that is feasible in original problem. These inequalities are often created by examining the current set of constraints, and using that information to create stronger constraints. The

theoretically strongest possible cutting planes are called facet-defining inequalities. The study of IPs and their facets is called Polyhedral Theory.

One method for creating strong cutting planes is through a process called lifting. The main idea behind lifting is to take a known valid inequality and alter one or more of its coefficients. By doing this intelligently, it is possible to create a valid cut that is stronger than the original inequality. Lifting is a topic often covered in literature [3, 6, 7, 24, 43], and lifting techniques have been shown to be useful for a wide range of applications, from general IPs to Graph Theory problems.

A graph is an abstract representation of a system. A graph consists of a set vertices and a set of edges. Vertices (also called nodes) typically represent entities in a system, while edges (or arcs) represent some relationship between the entities. For example, the vertices in a graph may represent cities on a map, while the edges represent a road between two cities. Two nodes are said to be adjacent if there is an edge connecting them.

The field of Graph Theory studies graphs, including various graph optimization problems. Popular Graph Theory problems include shortest route [5, 42], min-cost flow [18, 20], and graph coloring problems [11, 12]. Graph Theory problems have found applications in transportation [21, 23, 52], scheduling [15, 30], forecasting [45, 50], and even law enforcement [4]. Due to the discrete nature of the elements in a graph, Graph Theory problems are often modeled and optimized using IPs.

This thesis focuses on one Graph Theory problem, the Node Packing problem (al-

ternatively known as the Independent Set problem). The objective of this problem is to find the largest possible set of nodes, such that no two adjacent nodes are in the set. This problem is known to be $\mathcal{NP}$-Complete [32].

The Node Packing problem has been deployed in a diverse range of real-world situations, displaying its usefulness in practical applications. One such application is shown in a paper by Zwaneveld, et. al. [53]. This paper discusses a method for routing trains at a railway station. As one could imagine, there are many variables at play here (trains, entering and departing routes, etc.). Solving this routing problem is very important for the operators. An incorrect routing is not just bad for business, but a hazard to the well-being of train passengers. As the paper shows, the Node Packing problem is ideally suited to handle this logistical problem.

In other applications, the Node Packing problem has been deployed in genome mapping [27], machine scheduling [51], and sensor coverage [36]. Additionally, the concept of conflict graphs can be extended to other IPs [2], and much of the Polyhedral Theory developed on the Node Packing problem can be extended to these IPs as well.

1.1 Motivation

A common method for solving the Node Packing problem is to formulate it as an IP. The main motivation of this thesis is developing methods to find valid inequalities for these Node Packing IPs. There has been a significant amount of recent work in the area of simultaneous lifting and Polyhedral Theory at Kansas State University [10, 17, 28, 34].

Much of this work has been focused on knapsack problems, a special class of IP problems.

In 2009, Conley [14] extended this research to the Node Packing problem. The result was the uncovering of two previously undiscovered classes of graph structures, both of which generate valid inequalities for the Node Packing polytope. Both of these structures are combinations of previously known graph structures. This work left a new research question: can new inequalities be created from the combination of any general graph structures? One motivation for this thesis was to determine conditions under which this is possible, as well as a method for creating them.

While working on this question, a new method for lifting to an odd-hole inequality was discovered. While lifted odd-hole inequalities are not always helpful when solving general Node Packing problems, they have been shown to be useful under certain conditions [39]. The second aim of this research was to code and implement a this lifting technique to determine when it is most effective.

1.2 Contributions

This thesis outlines two new methods for creating valid inequalities for the Node Packing problem. The first method introduces the concept of a Simultaneous Lifting Expansion. In certain circumstances, this method allows for the combining of two separate valid inequalities into one, higher dimensional cut. The power in this method lies in the fact that it allows for any two inequalities to be combined. This method is shown to generalize Conley's work, and creates previously undiscovered classes of facet defining

inequalities for the Node Packing polyhedron.

The second contribution is a method for lifting from a common graph structure, the odd hole, to a facet defining inequality. Lifted odd-hole inequalities have been studied previously [38, 39], but this research introduces a new method for finding lifting coefficients, the Odd Gap Lifting (OGL) procedure. The exciting aspect of this procedure is that it can find exact lifting coefficients without having to solve a decision problem. The OGL procedure is a main component of the Cliques On Odd-holes Lifting (COOL) procedure. COOL intelligently orders nodes for lifting to the odd hole, resulting in a facet defining cut. A computational study was completed to test the COOL procedure against a typical odd-hole lifting strategy. This study found COOL to be most effective in graphs with a low edge density. COOL also performs well when the initial hole is small.

1.3 Outline

Chapter 2 provides definitions and background information necessary to understand the work done in this thesis. It provides the reader with a more rigorous understanding of the subjects mentioned in this chapter. Topics in Integer Programming, Polyhedral Theory, and Graph Theory are discussed.

Chapter 3 describes the lifting techniques derived for this thesis. The main results are Simultaneous Lifting Expansions, the Odd Gap Lifting procedure, and the Cliques On Odd-holes Lifting procedure. Relevant theorems are provided and proven for each

concept.

Chapter 4 describes a computational study testing the COOL procedure against a common odd-hole lifting scheme. The study's objectives and methodologies are outlined. Results are provided, along with an analysis of the findings.

Finally, Chapter 5 provides a conclusion of the thesis work. Important findings and concepts are discussed. Possible areas of future research are explored, along with suggestions for pursuing them.

Chapter 2

Background Information

This chapter outlines the background information needed to understand the work done for this thesis, including relevant concepts and definitions. The thesis draws from work done in Integer Programming, Polyhedral Theory, and Graph Theory. A particular focus is placed on lifting.

2.1 Integer Programming

Integer Programs (IPs) are optimization problems where decision variables are restricted to integer values. These problems are solved to either maximize or minimize some linear objective function, where the values of the variables are restricted by a number of constraints. Let $N = \{1, 2, ..., n\}$ contain the indices of the IP's variables. In general, IPs take the following form:

$$\begin{aligned} &\text{Maximize} \quad c^T x \\ &\text{subject to} \quad Ax \leq b \\ &\qquad\qquad x \in \mathbf{Z}_+^n. \end{aligned}$$

where $c \in \mathbf{R}^n$, $A \in \mathbf{R}^{m \times n}$ and $b \in \mathbf{R}^m$.

The most commonly used technique for solving IPs is called Branch and Bound. The first step in this method is to solve the linear relaxation of the IP. The linear relaxation problem is identical to the original IP, except the integer restrictions are taken away. If the linear relaxation solution contains one or more variable x_i at a non-integer value, one particular x_i is chosen. The linear relaxation then branches into two new child problems, both are identical to the linear relaxation with one added constraint. In particular, if $x_i = q$, one child problem adds the constraint $x_i \geq \lceil q \rceil$, and the other adds $x_i \leq \lfloor q \rfloor$. One child is chosen to be solved, and process continues. If a node is infeasible, solved to all integer values, or solved to an objective value worse than the best known integer solution, it is fathomed and no branching occurs. The process continues until all nodes are fathomed.

Unfortunately, Branch and Bound is computationally intensive. It requires solving a linear program for each new branch, and the number of branches can grow exponentially. For this reason, many techniques have been studied to try to improve solving time. One such method is to use cutting planes, which are studied in Polyhedral Theory.

2.2 Polyhedral Theory

Polyhedral Theory is a fundamental area of research in mathematical programming. This discipline studies the feasible space in integer and linear programs. This section addresses important definitions and results from Polyhedral Theory that are essential to this thesis.

A set S is said to be convex if and only if $(\lambda s_1 + (1-\lambda)s_2) \in S$ for all $s_1, s_2 \in S$ and $\lambda \in [0,1]$. The convex hull of some set R, denoted as $R^{ch} = conv(R)$, is the intersection of all convex sets S such that $R \subseteq S$.

The solution space of a linear inequality $\sum_{i=1}^{n} \alpha_i x_i \leq \beta$ is a half-space. Formally, the half space is the set $\{x \in \mathbf{R}^n : \sum_{i=1}^{n} \alpha_i x_i \leq \beta\}$. Clearly, this set is convex. A polyhedron is a finite intersection of half-spaces. A polytope is a bounded polyhedron.

Define $P = \{x \in \mathbf{Z}^n : Ax \leq b\}$ to be the set of all feasible points in an integer program. A well known result is that P^{ch} is a polyhedron. The feasible region of an IP's linear relaxation is denoted as $P^{LR} = \{x \in \mathbf{R}^n : Ax \leq b\}$. It is trivial to see that $P^{ch} \subseteq P^{LR}$.

Branch and Bound involves solving the linear relaxation of a problem, and any solution that is in P^{LR} but not in P is infeasible for the integer program. Therefore, a solution found in $P^{LR} \setminus P^{ch}$ may result in another set of branches. Clearly, removing some of this extra linear relaxation space would lessen the likelihood of a problem solving to a point that is not in P. This is the primary reasoning behind cutting planes.

A cutting plane is an inequality that is added to the set of constraints in the integer program. A cutting plane $\sum_{i=1}^{n} \alpha_i x_i \leq \beta$ is said to be valid if and only if $\sum_{i=1}^{n} \alpha_i x_i \leq \beta$ for all $x \in P$. In other words, there are no points in P that do not satisfy the inequality. Clearly, not all valid inequalities will be useful. The usefulness of a cut is often related to the dimension of the face that it induces on P^{LR}.

The face F induced by a cutting plane is the intersection of the inequality at equality with P^{ch}, or $F = \{x \in P^{ch} : \alpha^T x = \beta\}$. The dimension of this space is equal to the maximum number of linearly independent vectors found on the face. These vectors can be constructed by finding affinely independent points on the face.

A set of points $x_1, x_2, ..., x_d \in \mathbf{R}_+^n$ are affinely independent if and only if $\sum_{j=1}^{d} \lambda_j x_j = 0$ and $\sum_{j=1}^{d} \lambda_j = 0$ is solved uniquely by $\lambda_j = 0$ for $j = 1, 2, ..., d$. To find linearly independent vectors from a set of affinely independent points, select one x_i then calculate the vector between x_i and x_j for all $j = 1, 2, ..., d$ where $j \neq i$. This process defines $d-1$ linearly independent vectors in the space. Thus, the dimension of any face is one less than the maximum number of affinely independent points in the face.

A facet defining inequality induces a face on P^{ch} of dimension $dim(P^{ch}) - 1$. This is the strongest class of cutting planes. If an integer programming formulation contains every facet defining inequality, then $P^{ch} = P^{LR}$, and every basic solution to the linear relaxation is an integer solution. This would mean that no branching is necessary to solve the integer program.

To better illustrate the concepts of cutting planes and facet defining cuts, consider

the following integer program:

Maximize $x_1 + 2x_2$

Subject to $5x_1 + 4x_2 \leq 20$

$3x_1 + 5x_2 \leq 18$

$x_1,\ x_2\ \in \mathbf{Z}_+$.

Figure 2.2 provides a graphical view of this IP. Feasible solutions for P^{LR} lie in the area below both constraints in the first quadrant. All feasible integer solutions, the points in P, are represented in the graph by large circles. The dashed line defined by $x_1 + x_2 \leq 4$ is a valid cutting plane, as it is satisfied by every point in P. Further, it is a facet defining cut.

To show this, observe that P^{ch} is a two-dimensional space. The inequality $x_1+x_2 \leq 4$ is met at equality by two affinely independent points, (0,4) and (3,1), so it defines a face on P^{ch} of dimension 1. Since the inequality is valid and creates a face of size $dim(P^{ch})-1$, it must be facet defining.

This example illustrates the usefulness of cutting planes in solving IPs. Notice that the inequality eliminates all of the linear relaxation space in the triangle defined by points A, B, and C. This reduces the chances for a linear relaxation to solve at a non-integer point, meaning it is likely that less branching will occur while solving the problem.

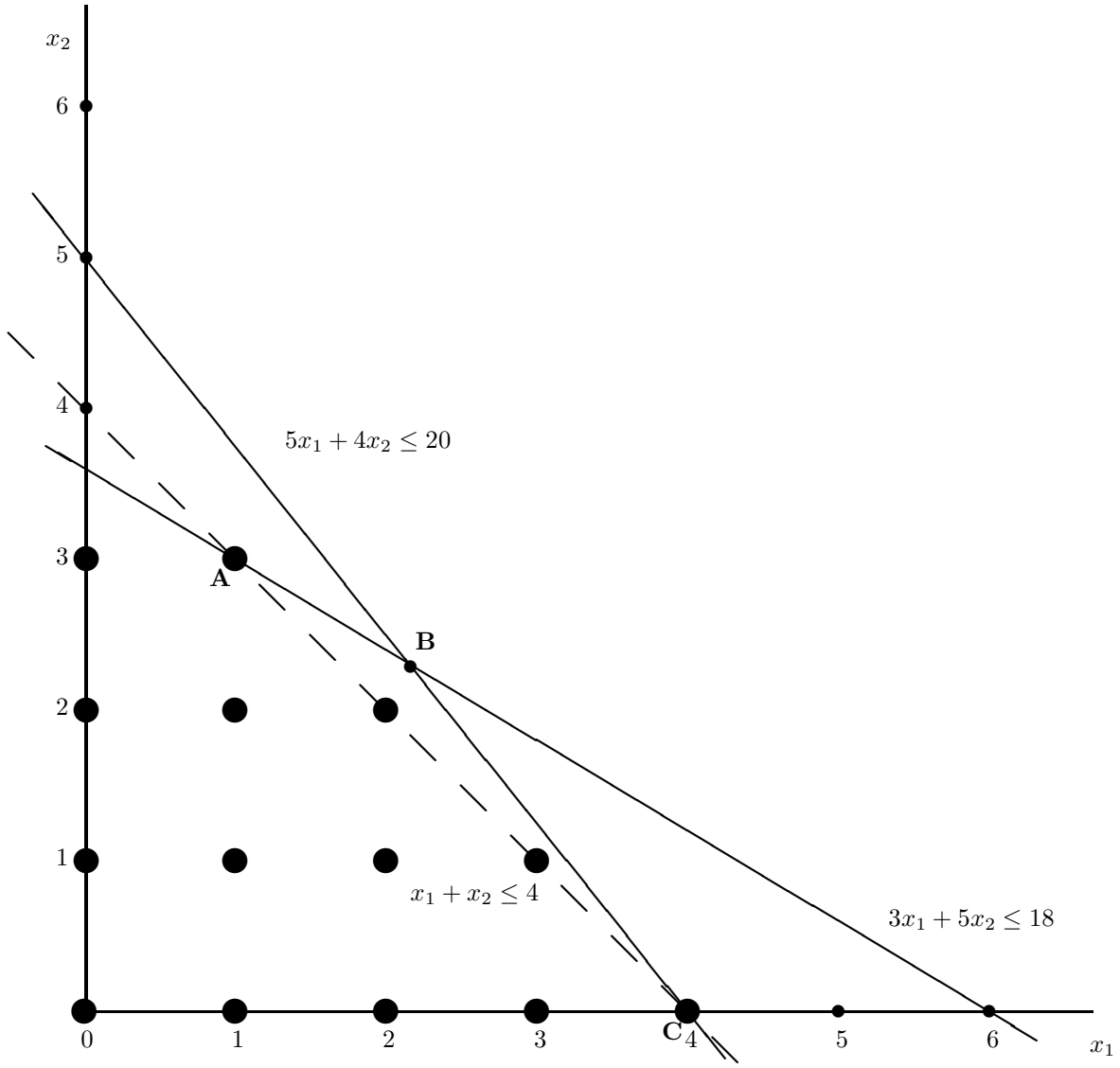

Figure 2.1: A facet defining cut [28]

2.3 Graph Theory

In mathematics, graphs are abstract representations of systems. They are used to highlight certain relationships between entities, and are often useful as visualizations of optimization problems. Their properties are studied in the field of Graph Theory.

A graph $G = (V, E)$ is a set of vertices V and a set of edges E. The elements of E are of the form $\{u, v\}$ where $u, v \in V$. In visual representations, vertices (also called nodes)

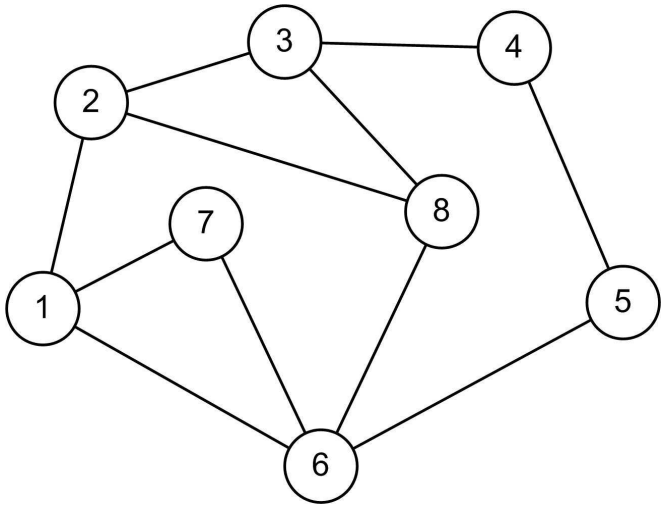

Figure 2.2: A sample graph

are usually drawn as circles. If $\{u, v\} \in E$, an edge (arc) is drawn as a line connecting vertices u and v. In this case, u and v are said to be adjacent. An example of a graph is given in Figure 2.2.

The degree of vertex i, denoted $deg(i) = |\{\{i, j\} : \{i, j\} \in E\}|$, is the number of edges incident to i in G. A walk is a set of vertices $(v_1, v_2, ...v_q)$ such that $\{v_i, v_i + 1\} \in E$ for $i = 1, 2, ..., q - 1$. Informally, a walk is a list of vertices such that it is possible to travel from one vertex to the next through a connecting arc. A path is a walk without any repeated vertices. A cycle is a path with the added condition that the last vertex in the path is also the starting vertex.

Using Figure 2.2 as an example, note that $deg(2) = 3$ and $deg(4) = 2$. An example of a walk is $(2, 3, 8, 2, 1)$, which has a length of 4. The set $(1, 7, 6, 8, 3)$ represents a path of length 4, and $(2, 8, 6, 7, 1, 2)$ is a cycle of size 5.

Many problems that are often set up and solved on graphs. The first famous example

of this was Euler's Seven Bridges of Königsberg problem [9]. Other examples include network flow problems [25, 35], the traveling salesman problem [8, 26], edge covering problems [37, 44], and graph coloring problems [11, 12]. This research focuses on the Node Packing problem, which is presented later in this chapter.

2.3.1 Classical Induced Subgraphs

Sometimes it is useful to consider only a section of a graph, called a subgraph. A graph $G' = (V', E')$ is a subgraph of $G = (V, E)$ if and only if $V' \subseteq V, E' \subseteq E$, and if $\{u, v\} \in E'$ then $u, v \in V'$. A subgraph is called an induced subgraph if and only if E' contains all edges $\{u, v\} \in E$ for all $u, v \in V'$. Various subgraph structures are often isolated and studied due to their special features in certain applications. The following paragraph defines some of these structures; the clique, the hole, the anti-hole, the wheel, and the star.

A clique with p nodes, denoted K_p, is a graph with edge set $E(K_p) = \{\{v_i, v_j\} : i, j \in V'\}$. A hole with p nodes, H_p, is a graph with edge set $E(H_p) = \{\{v_i, v_{(i \bmod p)+1}\} : i = 1, 2, ..., p\}$. An anti-hole on p nodes, A_p, is a graph with edge set $E(A_p) = E(K_p) \backslash E(H_p)$. A wheel on p nodes, W_p, has edge set $E(W_p) = E(H_{p-1}) \cup \{\{v_p, v_i\} : i = 1, 2, ..., p-1\}$. A star on p nodes, S_p, has the edge set $E(S_p) = \{\{v_i, v_p\} : i = 1, 2, ..., p-1\}$.

Examples of these structures can be seen in Figure 2.3. The subgraph induced by nodes 1-6 is a wheel of size 6, or a W_6. Nodes 7-10 induce a K_4, 11-15 induce an H_5, and 16-21 induce an A_6.

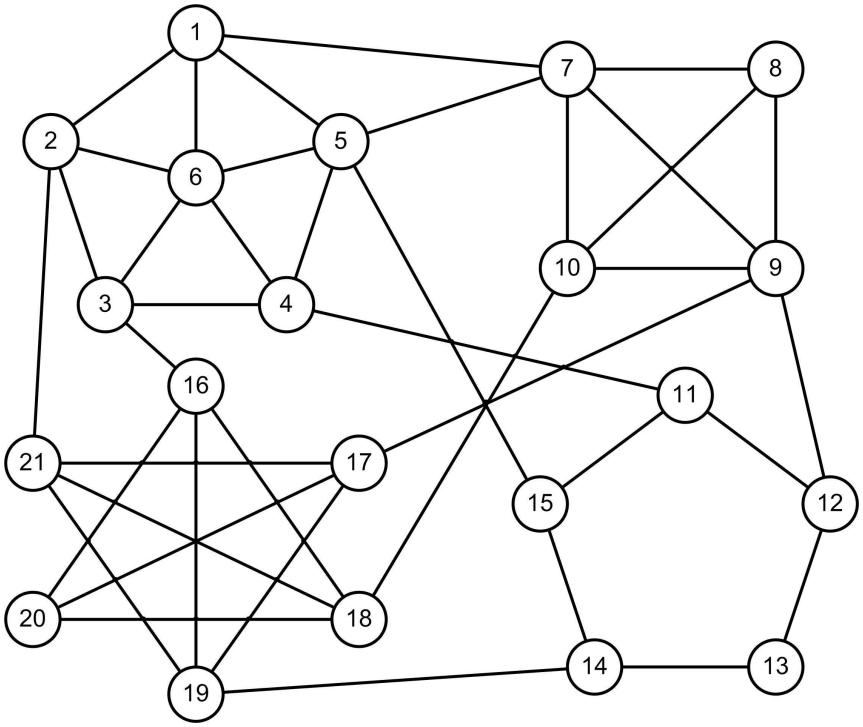

Figure 2.3: Induced subgraph examples

2.3.2 The Node Packing Problem

In a graph $G = (V, E)$, a node packing is a set of vertices $V' \subseteq V$ such that there is no edge $\{u, v\} \in E$ for any $u, v \in V'$. The cardinality Node Packing problem seeks to maximize $|V'|$. Alternately, in the weighted Node Packing problem, each vertex in $i \in V$ is assigned weight $w_i \in \mathbf{R}$, and the optimization problem seeks to find the set V' such that $\sum_{i \in V'} w_i$ is maximized.

The Node Packing problem is frequently solved as an IP. Each node i is assigned to an IP variable, x_i. If $x_i = 1$ in the IP, then node i is included in the packing. Otherwise, $x_i = 0$ and node i is not in the packing. The weighted Node Packing problem is formulated as an IP in the following manner:

$$\begin{aligned} \text{Maximize} \quad & \textstyle\sum_{i=1}^{n} w_i x_i \\ \text{subject to} \quad & x_i + x_j \leq 1 \quad \text{for all } \{i,j\} \in E \\ & x_i \in \{0,1\} \quad \text{for all } i \in V. \end{aligned}$$

The set of all feasible solutions to the IP is denoted as P^{NP}. Defining new facets of the convex hull of this set, P_{NP}^{ch}, is the focus of this thesis.

2.3.3 Induced Subgraph Inequalities for Node Packing

While the $x_i + x_j \leq 1$ inequalities are sufficient to fully define a Node Packing problem, there are more valid inequalities implicit in the structure of the graph. Many induced subgraphs generate specific valid inequalities for the Node Packing problem [41]. For example, take nodes 7-10 from Figure 2.3. These nodes induce a clique of size 4. Because all nodes in a clique are adjacent to each other, no packing can contain two members of the clique. Thus, if a K_p exists in the graph, the inequality $\sum_{i \in K_p} x_i \leq 1$ is valid in the IP formulation.

This clique inequality is also facet defining in P_{NP}^{ch} if the clique is maximal. For a graph with n vertices, the set of n affinely independent points is simple to obtain. Clearly, any point with $x_u = 1$ for some $u \in K_p$ and $x_i = 0$ for all $i \neq u$ will be feasible and meet the clique inequality at equality. Let the set A_1 contain each of these points, p in total.

Since the clique is maximal, for any $v \notin H_p$, there will be some $u \in H_p$ such that

$\{u, v\} \notin E$. This means that $\{u, v\}$ is a feasible packing, and the point with $x_u = 1, x_v = 1$ and $x_i = 0$ for all $i \neq u$ or v meets the clique inequality at equality. If set A_2 contains all of these points, then $A = A_1 \cup A_2$ contains n affinely independent points, showing that the inequality is facet defining.

A hole with p vertices has the associated valid inequality $\sum_{i=1}^{p} x_i \leq \lfloor \frac{p}{2} \rfloor$. This inequality is valid for any size hole, and it is facet defining on the induced subgraph if p is odd. This means that an H_5 has an associated odd-hole inequality of $\sum_{i=1}^{5} x_i \leq 2$, and this inequality defines a face on P_{NP}^{ch} of at least dimension 4. Figure 2.4 gives the five affinely independent points that show this.

A wheel of size p creates the W_p inequality $\lfloor \frac{p-1}{2} \rfloor x_p + \sum_{i=1}^{p-1} x_i \leq \lfloor \frac{p-1}{2} \rfloor$. This inequality is facet defining on the induced subgraph if p is even. So for a W_6, the inequality $2x_6 + \sum_{i=1}^{5} x_i \leq 2$, defines a face on P_{NP}^{ch} of at least dimension 5. Figure 2.4 gives the six affinely independent points to illustrate this.

Similarly, for an anti-hole A_p, the inequality $\sum_{i=1}^{p} x_i \leq 2$ is valid, and facet defining on the induced subgraph when p is odd. An A_5 creates the inequality $\sum_{i=1}^{5} x_i \leq 2$, and this inequality defines a face on P_{NP}^{ch} of at least dimension 4. Again, Figure 2.4 shows the five affinely independent points.

Unlike the maximal clique inequalities, the inequalities for wheels, odd-holes, and odd anti-holes are not facet defining for P_{NP}^{ch} in general. This thesis discusses a method for lifting onto the odd-hole inequality to create a facet defining cut for the entire graph.

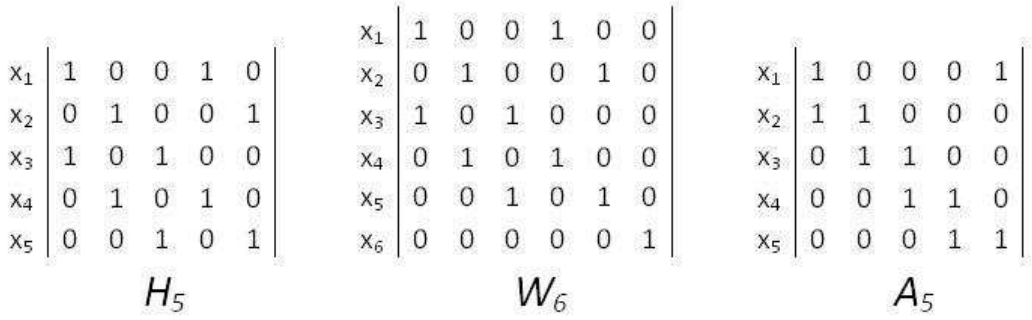

Figure 2.4: Affinely independent points for H_5, W_6, and A_5

2.3.4 Conflict Graphs

Many of the Polyhedral Theory results developed for the Node Packing problem can be extended to general binary IPs through conflict graphs [2, 40]. Conflict graphs are created from IPs by creating a node to represent each variable in the problem, and drawing edges between two nodes in conflict.

Let the conflict graph of a binary integer program be represented by $G = (V, E)$. Every variable x_i in the IP is represented by a node $i \in V$. The edge $\{i, j\}$ is in E if setting both x_i and x_j to 1 is infeasible in the IP. To illustrate this, consider the following binary IP:

$$\begin{aligned} \text{Maximize} \quad & 5x_1 + 7x_2 + 3x_3 + x_4 + 4x_5 \\ \text{subject to} \quad & 2x_1 + 3x_3 + 2x_5 \leq 4 \\ & 4x_1 + 2x_2 + x_5 \leq 5 \\ & x_1 + 2x_3 + 3x_4 \leq 3 \\ & x_i \in \{0, 1\} \quad \text{for all } i. \end{aligned}$$

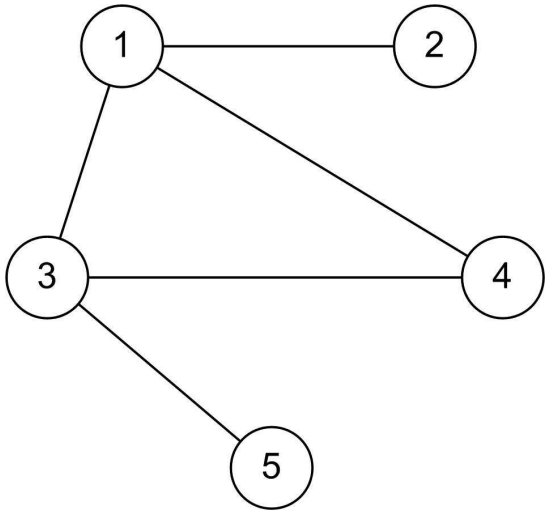

Figure 2.5: Conflict graph

Examining the constraints reveals relationships between variable values in a feasible solution. We can see from the first constraint that any solution where $x_1 = 1$ and $x_3 = 1$ is infeasible, since $2x_1 + 3x_3 = 5 > 4$. So the conflict graph contains an edge between nodes 2 and 3. Following this logic for all inequalities, the conflict graph in Figure 2.5 is obtained.

From this graph, one can deduce the Node Packing inequalities $x_1+x_2 \leq 1,\ x_1+x_3 \leq 1,\ x_1+x_4 \leq 1,\ x_3+x_4 \leq 1$, and $x_3+x_5 \leq 1$. Clearly, each of these constraints is valid for the IP. Observe that nodes 1, 3, and 4 induce a K_3, thus the inequality $x_1 + x_3 + x_4 \leq 1$ is valid. Further, since the clique is maximal, this inequality is facet defining for P^{ch}. Now consider the point $x = (0.5, 0, 0.5, 0.5, 0)$. This point is in P^{LR}, but is cut off by the clique inequality. This illustrates that conflict graphs can help create useful inequalities for binary IPs.

2.4 Lifting

Lifting is a technique used to strengthen a given inequality. It involves altering the coefficients in an inequality in such a way that cuts out more non-integer space. This technique takes a lower-dimensional inequality and builds up to a stronger, higher dimensional cut. The technique was first introduced by Gomory [22], and many advancements have been made since [3, 6, 7, 24, 43].

First, define the restricted space of P^{ch} on the set $D \subseteq N$ as $P^{ch}_{D,K} = conv\{x \in P : x_j = k_j \text{ for all } j \in D\}$ where $k_j \in \mathbf{Z}$ and $K = (k_1, k_2, ..., k_{|D|})$. That is to say, the restricted space only considers a subset of the variables in the problem. Each variable with an index in D is forced to take on a fixed value, k_j.

Let $\sum_{j\in D} \alpha_j x_j + \sum_{j\in N\setminus D} \alpha_j x_j \leq \beta$ be a valid inequality for $P^{ch}_{D,K}$. A lifting procedure creates an inequality of the form $\sum_{j\in D} \alpha'_j x_j + \sum_{j\in N\setminus D} \alpha_j x_j \leq \beta'$ which is valid over P^{ch}. Lifting techniques are classified based on how coefficients are changed (up, down, or middle lifting), how many variables are considered (sequential versus simultaneous lifting), how strong the resulting cuts are (exact versus approximate lifting), and the number of inequalities obtain (single or synchronized lifting).

In up lifting, every element in K is equal to zero, and the right-hand side β is not altered in the lifted inequality. For down lifting, each element in K is set to its upper bound, and the value of β is often decreased. There is also a middle lifting technique, which is a combination of the other two.

Distinctions of sequential or simultaneous refer to how many variables are being lifted. For sequential lifting, one variable is lifted at a time, thus $|D| = 1$. Often, this type of lifting is sequence dependent, where the coefficients of previously lifted variables can have an effect on the coefficient of the next variable. For simultaneous lifting, $|D| \geq 2$, and each variable in the set D is lifted at the same time.

Exact lifting procedures yield an inequality that is as strong as possible, such that any increase in α' or decrease in β' would make the inequality invalid. However, finding an exact lifting coefficient is often difficult and may require solving a separate optimization problem. In order to avoid the computational burden of finding exact lifting coefficients, several approximate lifting techniques have been developed. These are faster, heuristic approaches to determining lifting coefficients. While they may not yield the strongest possible inequalities, the resulting inequalities are guaranteed to be valid and are often strong enough to be effective [3, 6, 24].

To help illustrate the concept of lifting, consider the cardinality Node Packing problem on the graph in Figure 2.6. Notice that nodes 1-7 induce an H_7, so $\sum_{i=1}^{7} x_i \leq 3$ is valid for P_{NP}^{ch} and facet defining for the induced subgraph. To find an exact single sequential up-lifting coefficient for node 8, solve the following IP:

$$\begin{aligned}
&\text{Maximize} \quad x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 \\
&\text{subject to } x_i + x_j \leq 1 \quad \text{for all } \{i, j\} \in E \\
&\qquad\qquad x_8 = 1 \\
&\qquad\qquad x_i \in \{0, 1\} \quad \text{for all } i.
\end{aligned}$$

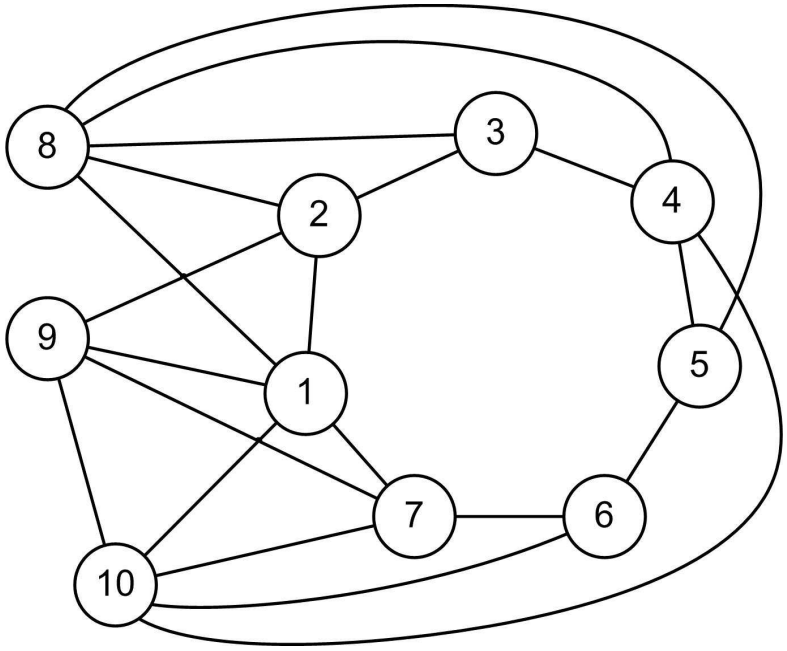

Figure 2.6: Node packing graph for lifting example

This formulation looks for the largest node packing that includes x_8. If the optimal objective value is Z, then $\alpha_8 = \beta - Z$. In this example, if node 8 is forced to be in the node packing, then of nodes 1-7, at most one can be in the packing (namely, either 6 or 7). So α_8 is equal to $3 - 1 = 2$. Now, the lifted inequality is $\sum_{i=1}^{7} x_i + 2x_8 \leq 3$. Moving on to node 9, a new IP is set up to determine the maximum value of $\sum_{i=1}^{7} x_i + 2x_8$ if x_9 is forced to 1. In this case, $\alpha_9 = 0$, because nodes 8 and 6 create a valid node packing whose Z value is 3. Node 10 is lifted in similarly, with coefficient $\alpha_{10} = 1$.

When this procedure is complete, the inequality $\sum_{i=1}^{7} x_i + 2x_8 + x_{10} \leq 3$ is obtained. When determining the lifting coefficient for each point, the IP solution reveals a new point on the face induced by the lifted inequality. Each new point is affinely independent to the others since it contains a new variable. Thus, the dimension of the inequality is increased by one with each new variable lifted. Figure 2.7 shows a matrix containing

x_1	1	0	0	1	0	1	0	0	0	0
x_2	0	1	0	0	1	0	1	0	0	1
x_3	1	0	1	0	0	1	0	0	0	0
x_4	0	1	0	1	0	0	1	0	0	0
x_5	1	0	1	0	1	0	0	0	0	1
x_6	0	1	0	1	0	1	0	0	1	0
x_7	0	0	1	0	1	0	1	1	0	0
x_8	0	0	0	0	0	0	0	1	1	0
x_9	0	0	0	0	0	0	0	0	1	0
x_{10}	0	0	0	0	0	0	0	0	0	1

Figure 2.7: Affinely independent points for lifting example

the 10 affinely independent points from this example. The first seven columns are the independent points from the induced H_7. The next three points are the solutions to the IPs used to determine α_8, α_9, and α_{10}, respectively.

It is also worth noting that, because the variables were lifted sequentially, a different lifting order may create a different facet defining inequality. For example, if node 9 had been lifted first, followed by 8 then 10, the resulting inequality would have been $\sum_{i=1}^{10} x_i \leq 3$.

Simultaneous lifting is less common than sequential lifting in Node Packing problems, but has been shown to be possible for certain structures. One example is the odd bipartite hole presented by Conley [14]. An odd bipartite hole is a graph $G = (V, E)$ such that V can be partitioned into two sets H_p and H_q which both induce odd holes. Further, E must contain every edge $\{i, j\}$ with $i \in H_p$ and $j \in H_q$. This graph is denoted as $OBP_{p,q}$. An $OBP_{5,7}$ can be seen in Figure 2.8.

This structure is nearly identical to a complete bipartite graph, but with the excep-

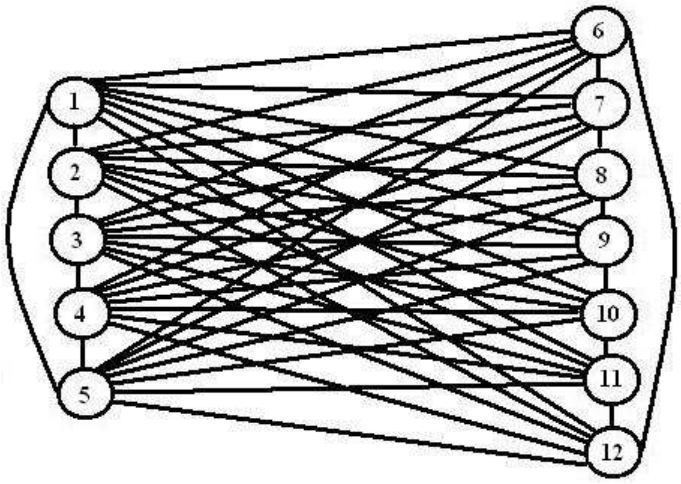

Figure 2.8: An odd bipartite hole [14]

tion that the nodes in each of the bipartitions induce odd holes. In this structure, it is clear that a valid node packing may not contain nodes from both odd holes. From this fact, Conley deduces the following valid inequality for the structure, which is also facet defining in the restricted space: $\sum_{i \in H_p} x_i + \sum_{i \in H_q} \frac{p-1}{q-1} x_i \leq \lfloor \frac{p}{2} \rfloor$.

If a node packing contains nodes in H_p, it may only contain $\lfloor \frac{p}{2} \rfloor$ nodes, thus the inequality is valid. If the node packing contains nodes in H_q, it will only contain $\lfloor \frac{q}{2} \rfloor$ nodes. To keep the inequality valid, this is multiplied by $\frac{\lfloor \frac{p}{2} \rfloor}{\lfloor \frac{q}{2} \rfloor} = \frac{p-1}{q-1}$. Thus, the inequality is valid for P_{NP}^{ch}.

This odd bipartite hole was one of the inspirations for the research in this thesis. Where the odd bipartite hole combines two holes to create a new inequality, this thesis presents a method for combining two general inequalities.

Chapter 3

Lifting in the Node Packing Polyhedron

This chapter provides the main results of the thesis. The main topics are Simultaneous Lifting Expansions, the Odd Gap Lifting procedure, and the Cliques On Odd-holes Lifting procedure. Each result is defined and relevant theorems are provided. For clarity, examples for each topic are included.

3.1 Simultaneous Lifting Expansions in the Node Packing Polyhedron

The main focus of this research was to develop new methods for developing cutting planes for the Node Packing problem. One inspiration for this research was work done by

Conley [14]. The focus of that paper is finding previously undiscovered classes of graph structures that allow for simultaneous lifting in a Node Packing problem. Of particular interest was how the paper was able to define inequalities for subgraphs that contain combinations of classical graph structures, such as cliques and holes. One motivation for this paper was to extend these findings to discover conditions when any two graph structures could be combined to create a new inequality.

The result of this research is the Simultaneous Lifting Expansion (SLE). It requires two valid inequalities for a Node Packing problem, with both inequalities sharing exactly one node. Further, for each node in one inequality, the adjacencies to the other inequality must be identical. In this situation, a method for combining the two original inequalities to generate a new valid inequality is presented. The dimension of this new cut is equal to the sum of the dimensions of the first two inequalities.

Prior to providing the major result of this section, the formal definition of a Simultaneous Lifting Expansion is required. Given a set of nodes $D \subset V$, the set $D' \subseteq V \setminus D$ is said to be an SLE of D if there exists a u in D such that for all $i \in D$, $\{u, i\} \in E$ if and only if $\{v, i\} \in E$ for all $v \in D'$.

In other words, u and every member of D' are adjacent to the same members of D. So, for any $v \in D'$, the subgraph induced by D is identical to the subgraph induced by $(D \cup \{v\}) \setminus \{u\}$. For an example, refer to Figure 3.1. Let $D = \{1, 2, 3, 4, 5\}$ and $D' = \{6, 7, 8, 9, 10\}$. Here, D' is an SLE for D, with $u = 5$. Note that the nodes in D induce an odd-hole, as does $D \setminus \{u\} \cup \{v\}$ for all $v \in D'$. In addition, $D' \cup \{5\}$ induces

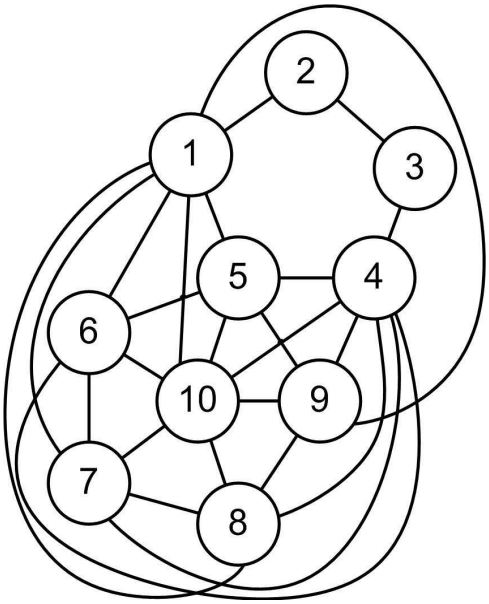

Figure 3.1: An example Simultaneous Lifting Expansion

a wheel, making this structure an SLE of a wheel on a hole. In such a situation, the following theorem provides the valid SLE inequality.

Theorem 3.1.1 *Let $D \subset V$ and D' be a simultaneous lifting expansion of D in regards to node u. Furthermore, let $\sum_{i \in D} \alpha_i x_i \leq \beta$ and $\sum_{i \in D' \cup \{u\}} \alpha'_i x_i \leq \beta'$ be valid inequalities of P^{ch}_{NP}. Then the Simultaneous Lifting Expansion inequality $\sum_{i \in D \setminus \{u\}} \alpha_i x_i + \frac{\alpha_u}{\beta'} \sum_{i \in D' \cup \{u\}} \alpha'_i x_i \leq \beta$ is a valid inequality of P^{ch}_{NP}.*

Proof: Let $x \in P^{NP}$. Clearly $\sum_{i \in D \setminus \{u\}} \alpha_i x_i + \alpha_u x_u \leq \beta$ and $\sum_{i \in D' \cup \{u\}} \alpha'_i x_i \leq \beta'$ since they are valid inequalities. Furthermore $\frac{1}{\beta'} \sum_{i \in D' \cup \{u\}} \alpha'_i x_i \leq 1$. Since $x_u \leq 1$, and all members of $D' \cup \{u\}$ are adjacent to the same members of $D \setminus \{u\}$, one obtains $\sum_{i \in D \setminus \{u\}} \alpha_i x_i + \frac{\alpha_u}{\beta'} \sum_{i \in D' \cup \{u\}} \alpha'_i x_i \leq \beta$. Thus, the SLE inequality is valid.

□

Recall the example in Figure 3.1, with $D = \{1, 2, 3, 4, 5\}$, $D' = \{6, 7, 8, 9, 10\}$, and $u = 5$. Notice that D induces an odd-hole, so the inequality $\sum_{i=1}^{5} x_i \leq 2$ is valid for P_{NP}^{ch}. Also, $D' \cup \{5\}$ induces a wheel, with valid inequality $2x_{10} + \sum_{i=5}^{9} x_i \leq 2$. So the SLE inequality $\sum_{i=1}^{4} x_i + \frac{1}{2}(2x_{10} + \sum_{i=5}^{9} x_i) \leq 2$ is valid for P_{NP}^{ch}.

While having a valid inequality is certainly important, it is more useful to know the dimension of the face it induces. The next theorem provides a lower bound for the dimension of the SLE inequality. Prior to providing this result, let ξ_i be the point in $\mathbf{R}^n$ created by the i^{th} column of an $n \times n$ identity matrix.

Theorem 3.1.2 *Let $D \subset V$ and $D' \subseteq V \setminus D$ be a simultaneous lifting expansion of D in regards to node u. If $\sum_{i \in D} \alpha_i x_i \leq \beta$ defines a face of dimension r in P_D^{NP}, $\sum_{i \in D' \cup \{u\}} \alpha'_i x_i \leq \beta'$ defines a face of dimension s in $P_{D' \cup \{u\}}^{NP}$ and the face in P_D^{NP} contains at least one point with $x_u = 1$, then the SLE inequality $\sum_{i \in D \setminus \{u\}} \alpha_i x_i + \frac{\alpha_u}{\beta'} \sum_{i \in D' \cup \{u\}} \alpha'_i x_i \leq \beta$ defines a face of dimension at least $r + s$ in P^{NP}.*

Proof: By assumption, there exists a set of $r + 1$ affinely independent points, B_D, in P^{NP} where $\sum_{i \in D} \alpha_i x_i = \beta$ for all $x \in B_D$. Additionally, there exists a set of $s+1$ affinely independent points, $B_{D' \cup \{u\}}$, in $P_{D' \cup \{u\}}^{NP}$ where $\sum_{i \in D' \cup \{u\}} \alpha'_i x_i = \beta'$ for all $x \in B_{D' \cup \{u\}}$. Let y be some point in B_D where $x_u = 1$ and y' be any point in $B_{D'}$.

Define $B_1 = \{x : x \in B_D,\ x_u = 0\} \cup \{(x - \xi_u) + y' : x \in B_D,\ x_u = 1\}$ and $B_2 = \{x + (y - \xi_u) : x \in B_{D' \cup \{u\}} \setminus \{y'\}\}$. Let $B = B_1 \cup B_2$, so B has $r + s + 1$ columns. Figure 3.2 shows the format of B, where X is a $|D' \cup \{u\}| \times |D' \cup \{u\}|$ matrix whose columns contain either all zeros or $y'|_{D' \cup \{u\}}$ depending upon the value of x_u in

$$B = \left| B_1 \quad B_2 \right| = \left| \begin{array}{c|cc} x_1 & & \\ \vdots & B_D|_{D\setminus\{u\}} & y|_{D\setminus\{u\}} \\ x_{|D\setminus\{u\}|} & & \\ x_{|D\setminus\{u\}|+1} & & \\ \vdots & X & B_{D'}|_{D'\cup\{u\}} \\ x_{r+s+1} & & \end{array} \right|$$

Figure 3.2: Form of the SLE B matrix

the corresponding point in B_D.

To show that B is affinely independent, let B_2' be the matrix created by subtracting the column in B equal to $(y - \xi_u + y')$ from every column in the B_2 matrix. Let B', shown in Figure 3.3, be $B_1 \cup B_2'$. This makes every entry in the upper-right portion of the B matrix zero and every entry in the lower-right portion equal to $x - y'$ for $x \in B_{D'\cup\{u\}}$ and $x \neq y'$. Since the points in $B_{D'\cup\{u\}}$ are affinely independent, the points $x - y'$ for $x \in B_{D'\cup\{u\}}$ and $x \neq y'$ are linearly independent.

By assumption, the r points in the upper-left portion B' are affinely independent. They are also independent from the points on the right, due to the zero entries in the upper right portion. Further, each point of these $r + s + 1$ points clearly meet the SLE inequality at equality. Thus this inequality has dimension at least $r + s$ and the result follows.

□

As an example, refer back to the discussion of the graph given in Figure 3.1. Recall that $D' = \{6, 7, 8, 9, 10\}$ is an SLE of $D = \{1, 2, 3, 4, 5\}$ in regards to node 5, and the SLE inequality for this structure is $\sum_{i=1}^{4} x_i + \frac{1}{2}(2x_{10} + \sum_{i=5}^{9} x_i) \leq 2$. Figure 3.4 shows the

$$B' = \left[\begin{array}{c|cc} x_1 & & \\ \vdots & B_D|_{D\setminus\{u\}} & 0 \\ x_{|D\setminus\{u\}|} & & \\ \hline x_{|D\setminus\{u\}|+1} & & \\ \vdots & X & B_{D'} - y'|_{D'\cup\{u\}} \\ x_{r+s+1} & & \end{array}\right]$$

Figure 3.3: Form of the SLE B' matrix

$$B = \left[\begin{array}{cccccc:cccccc} 1 & 0 & 0 & 1 & 0 & & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & & 0 & 0 & 0 & 0 & 1 \end{array}\right]$$

Figure 3.4: B matrix for SLE example

B matrix for this structure. All 10 points in the matrix meet the inequality at equality.

The B_D and $B_{D'\cup\{u\}}$ matrices, which contain the affinely independent points for the original structures, are found in Figure 3.5. Note that B_D contains five affinely independent points which define a face of dimension four in P_D^{NP}, while $B_{D'\cup\{u\}}$ contains six affinely independent points defining a face of dimension five in $P_{D'\cup\{u\}}^{NP}$. Let the point y be the third column in the B_D matrix, and let y' be defined by the second column in the $B_{D'\cup\{u\}}$ matrix.

The points to the left of the dashed line in Figure 3.4 define the matrix B_1. For these points, the top 4 entries are identical to the B_D matrix. If the fifth entry was equal to

$$B_D = \begin{vmatrix} 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{vmatrix} \qquad B_{D' \cup \{u\}} = \begin{vmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{vmatrix}$$

Figure 3.5: B_D and $B_{D' \cup \{u\}}$ for SLE example

zero in the B_D matrix, the rest of the entries in the column are zeros. If the fifth entry was equal to one, then the final five entries come from point y'.

For the B_2 matrix (the final five points in B), the first four entries come from y. The final six entries are identical to one column in the $B_{D' \cup \{u\}}$ matrix. Note that the column in $B_{D' \cup \{u\}}$ associated with y' was excluded, because this column would have been a duplicate of the third column in B_1 (the column associated with y).

The B_2' matrix is created by subtracting the column associated with y from every column in B_2. The union of B_2' and B_1 defines B'. Figure 3.6 shows this matrix partitioned into four quadrants. Due to the presence of the zeros in the upper-right quadrant, it is clearly evident that the five points on the left are independent from the five on the right. Since each of the five leftmost points came from B_D, an independent matrix, they are affinely independent. Similarly, the five points on the right all come from $B_{D' \cup \{u\}}$. The only alteration to these points was the subtraction of y', which also came from $B_{D' \cup \{u\}}$. Thus all of these points must be affinely independent.

$$B' = \left[\begin{array}{cccccc:cccccc}
1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hdashline
0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 & -1 & -1 & -1 & 0 & -1 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 & -1 & -1 & 0 & -1 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{array}\right]$$

Figure 3.6: B' for SLE example

Since there are ten affinely independent points meeting the SLE inequality at equality, this inequality must define a face of P_{NP}^{ch} of at least dimension 9. This is exactly $r+s$, as the theorem states. The results of this example illustrate the following corollary, which follows directly from Theorem 3.1.2:

Corollary 3.1.3 *Let $D \subset V$ and $D' \subseteq V \setminus D$ be a simultaneous lifting expansion of D in regards to node u. If $\sum_{i \in D} \alpha_i x_i \leq \beta$ is facet defining for P_D^{NP} and $\sum_{i \in D' \cup \{u\}} \alpha'_i x_i \leq \beta'$ is facet defining for $P_{D' \cup \{u\}}^{NP}$, then the SLE inequality $\sum_{i \in D \setminus \{u\}} \alpha_i x_i + \frac{\alpha_u}{\beta'} \sum_{i \in D' \cup \{u\}} \alpha'_i x_i \leq \beta$ is facet defining for $P_{D \cup D'}^{NP}$*

□

In addition, the SLE inequality can be facet defining for P_{NP}^{ch} when the conditions of the next corollary are fulfilled.

Corollary 3.1.4 *Let $D \subset V$ and $D' \subseteq V \setminus D$ be a simultaneous lifting expansion of D in regards to node u. If $\sum_{i \in D} \alpha_i x_i \leq \beta$ defines a face of dimension r in P_D^{NP},*

$\sum_{i \in D' \cup \{u\}} \alpha'_i x_i \leq \beta'$ *defines a face of dimension* s *in* $P^{NP}_{D' \cup \{u\}}$ *and* $r + s + 1 = |V|$*, then the SLE inequality* $\sum_{i \in D \setminus \{u\}} \alpha_i x_i + \frac{\alpha_u}{\beta'} \sum_{i \in D' \cup \{u\}} \alpha'_i x_i \leq \beta$ *is facet defining for* P^{ch}_{NP}*.*

□

3.1.1 SLE Examples

In his paper [14], Conley introduces two new structures, the odd bipartite hole and the cliqued hole. The valid inequality for both structures can be derived using SLE.

Recall the odd bipartite hole discussed in chapter 2. An odd bipartite hole is a graph $G = (V, E)$ such that V can be partitioned into two sets H_p and H_q which both induce odd holes. Further, E must contain every edge $\{i, j\}$ with $i \in H_p$ and $j \in H_q$. This graph is denoted as $OBP_{p,q}$. An $OBP_{5,7}$ can be seen in Figure 2.8.

To derive the SLE inequality for the $OBP_{5,7}$, let $D = \{1, 2, 3, 4, 5, 6\}$, $D' = \{7, 8, 9, 10, 11, 12\}$, and $u = 6$. Note that the nodes in D create a wheel, while $D' \cup \{u\}$ creates an odd-hole. For a general case, if u is some element of H_q, then $D = \{i : i \in H_p\} \cup \{u\}$ and $D' = \{j : j \in H_q \setminus \{u\}\}$. From this, we can see that the SLE inequality for the $OBH_{p,q}$ is $\sum_{i \in H_p} x_i + \frac{\lfloor \frac{p}{2} \rfloor}{\lfloor \frac{q}{2} \rfloor} \sum_{j \in H_q} x_j \leq \lfloor \frac{p}{2} \rfloor$.

Conley's other primary structure is the cliqued hole. This is a structure with m cliques such that if one vertex is selected from each of the m cliques, the resulting induced subgraph graph is a hole. Formally, a graph $G = (V, E)$ is a cliqued hole if and only if V can be partitioned into m sets, $P_1, P_2, ..., P_m$ such that $E = \cup_{i=1}^{m} \{\{v, w\} :$

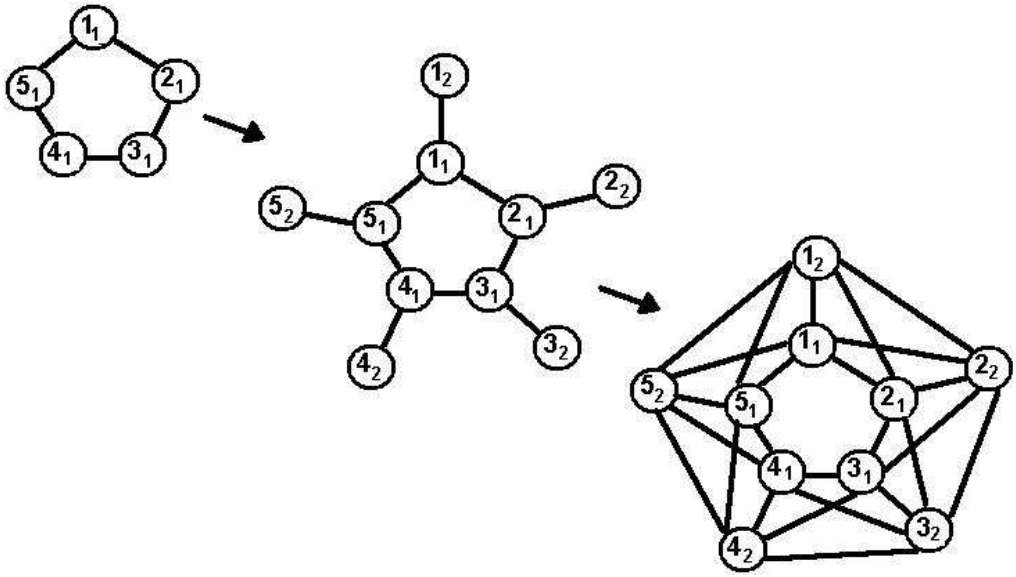

Figure 3.7: Cliqued hole transition [14]

$v, w \in P_i\} \bigcup \cup_{i=1}^{m} \{\{v, w\} : v \in P_i, w \in P_{(imodm)+1}\}$. Such a graph is denoted $CH_{m,P}$ where P is a vector containing the size of each clique in the graph. Figure 3.7 shows a transition from a hole to a cliqued hole for a $CH_{5,P}, P = (2, 2, 2, 2, 2)$.

In this structure, the SLE procedure can be applied iteratively to derive Conley's cliqued hole inequality. In this case, D consists of the nodes that make up the hole, while D' is made from the nodes in one of the cliques. Recall that the SLE inequality is of the form $\sum_{i \in D \setminus \{u\}} \alpha_i x_i + \frac{\alpha_u}{\beta'} \sum_{i \in D' \cup \{u\}} \alpha'_i x_i \leq \beta$. Since D induces a hole, $\alpha_u = 1$, and since $D' \cup \{u\}$ induces a clique, $\beta' = 1$ and $\alpha'_i = 1$ for all $i \in D' \cup \{u\}$. Because of this, the inequality becomes $\sum_{i \in D \cup D'} x_i \leq \lfloor \frac{m}{2} \rfloor$. Repeating the same procedure for each clique in the $CH_{m,P}$, the final inequality becomes $\sum_{i \in V} x_i \leq \lfloor \frac{m}{2} \rfloor$.

For both of these structures, the SLE results agree with Conley's findings. This illustrates that SLEs generalize Conley's work. In addition, SLEs allow for the creation

of a wide range of valid inequalities in seemingly chaotic graph structures. Consider the following example from the graph shown in Figure 3.8.

This graph exhibits multiple SLEs. First, consider the H_5 induced by nodes 1-5. Node 6 may be lifted to the H_5 inequality to obtain $\sum_{i=1}^{6} x_i \leq 2$. Because nodes 7-11 have the same adjacencies as node 6, they are an SLE of nodes 1-6 in regards to node 6. Nodes 6-11 induce a W_6 with inequality $\sum_{i=6}^{10} x_i + 2x_{11} \leq 2$. So $\sum_{i=1}^{5} x_i + \frac{1}{2}(2x_{11} + \sum_{i=6}^{10} x_i) \leq 2$ is a valid inequality.

Next, notice that nodes 12-17, together with node 5, induce an H_7 with inequality $x_5 + \sum_{i=12}^{17} x_i \leq 3$. Nodes 12-17 are an SLE of nodes 1-11 in regards to node 5, thus $\sum_{i=1}^{4} x_i + \frac{1}{2}(2x_{11} + \sum_{i=6}^{10} x_i) + \frac{1}{3}(x_5 + \sum_{i=12}^{17} x_i) \leq 2$ is valid.

Finally, nodes 17-21 induce an A_5 with inequality $\sum_{i=17}^{21} x_i \leq 2$. Nodes 18-21 are an SLE of nodes 1-17 in regards to node 17. So, the final SLE inequality is $\sum_{i=1}^{4} x_i + \frac{1}{2}(2x_{11} + \sum_{i=6}^{10} x_i) + \frac{1}{3}(x_5 + \sum_{i=12}^{16} x_i) + \frac{1}{6}(\sum_{i=17}^{21} x_i) \leq 2$ is valid. Further, each SLE inequality was created from two structures that were facet defining on their induced subgraphs. Thus, the final inequality is facet defining for P_{NP}^{ch}. Figure 3.9 shows a set of 21 affinely independent points that meet this inequality at equality, confirming that the cut is indeed facet defining. These points were constructed in the manner described in Theorem 3.1.2 for creating the B matrix, applied iteratively in the order that the sets were lifted.

Prior to this research, the inequality $\sum_{i=1}^{4} x_i + \frac{1}{2}(2x_{11} + \sum_{i=6}^{10} x_i) + \frac{1}{3}(x_5 + \sum_{i=12}^{16} x_i) + \frac{1}{6}(\sum_{i=17}^{21} x_i) \leq 2$ would not have been obtainable by any other obvious lifting procedure.

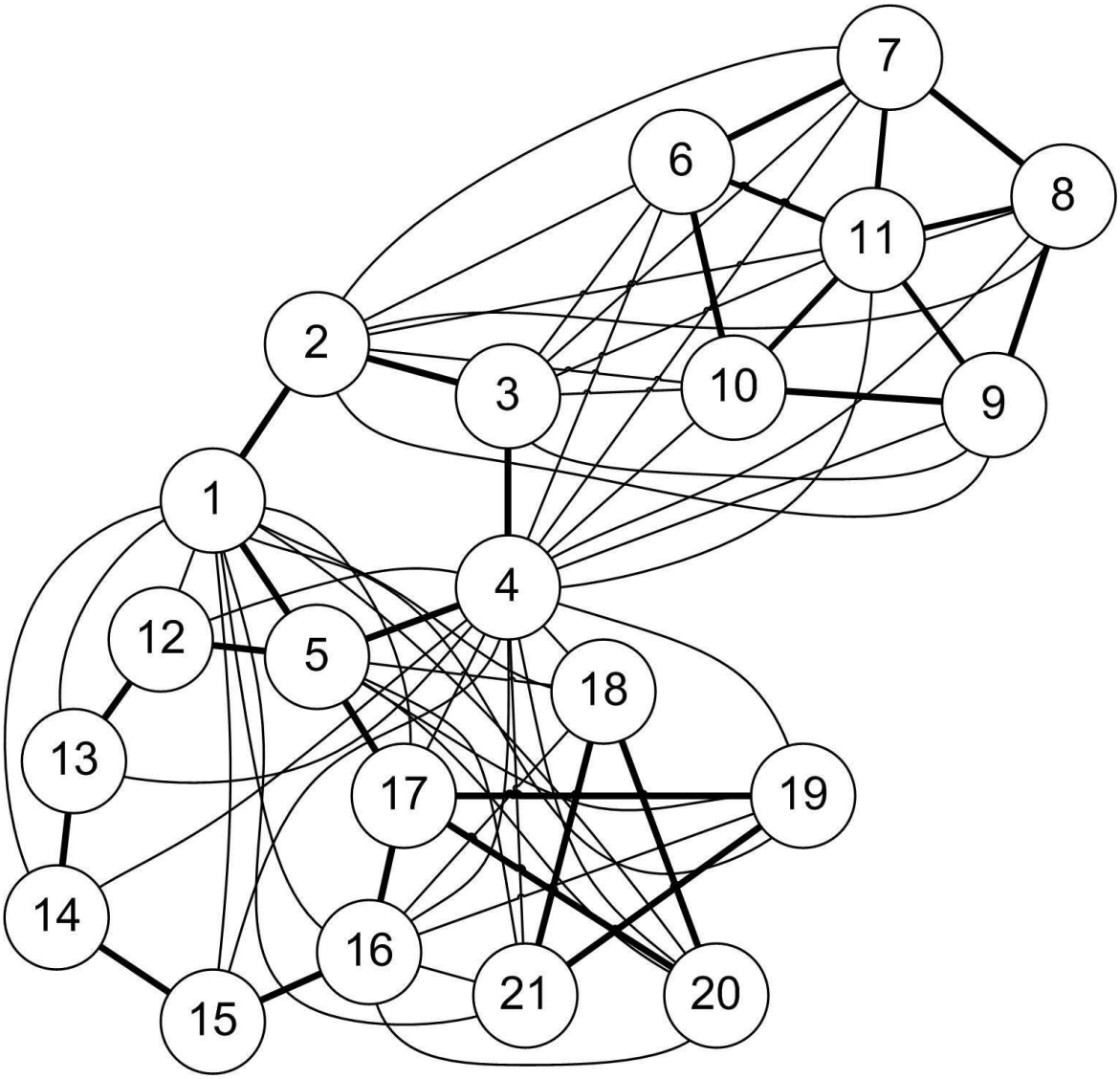

Figure 3.8: A larger SLE example

x_1	1	0	0	1	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
x_2	0	1	0	0	1	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
x_3	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
x_4	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
x_5	0	0	1	0	1	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0
x_6	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
x_7	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
x_8	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
x_9	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0
x_{10}	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0
x_{11}	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
x_{12}	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	1	1	1	1	1
x_{13}	0	0	1	0	1	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0
x_{14}	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	1	1	1	1	1
x_{15}	0	0	1	0	1	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0
x_{16}	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0
x_{17}	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	0	0	1
x_{18}	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1	1	0	0	0
x_{19}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0
x_{20}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0
x_{21}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

Figure 3.9: Affinely independent points for SLE example

Observe that the initial inequality with x_6 sequentially lifted was $\sum_{i=1}^{6} x_i \leq 2$. In any sequential or simultaneous up lifting technique, no coefficient in the initial valid inequality is modified. Since x_5 decreases from $\alpha_5 = 1$ to $\alpha'_5 = \frac{1}{3}$, no up lifting technique can be used to generate it. Furthermore, many coefficients have moved from $\alpha_i = 0$ to $\alpha'_i > 0$ so several variables are up lifted.

In some bizarre sense, SLE is performing both simultaneous up and down lifting at the same time. Furthermore, it presents how to merge two valid inequalities in a lifting procedure. To the best of the author's knowledge, this is the first such example

presented in the literature. Consequently, this method creates previously undiscovered classes of facet defining inequalities.

3.2 Sequential Lifting to a Hole Inequality

Odd-holes are a common graph structure that manifest in many Node Packing problems. However, their associated Node Packing inequality is not facet defining in general for P_{NP}^{ch}. This section describes a sequential lifting method that creates a facet defining cut from an odd-hole inequality. This method is based on the "gaps" between node v's adjacencies in the hole. Before providing the results of this section, several definitions are needed.

Let H_p be a hole with vertices $v_1, ..., v_p$. Let $v \in V \setminus V(H_p)$. Let the vertices $v_{j_1}, v_{j_2}, ..., v_{j_q} \in V(H_p)$ be adjacent to v with ascending indices in H_p. Denote the size of the j_k gap as $s_{j_k} = (j_{(k \ mod \ q)+1} - j_k) \ mod \ p$. If s_{j_k} is odd, then the j_k gap is said to be an odd gap with $o_{j_k} = 1$, and $t_{j_k} = 0$. If s_{j_k} is even, then the j_k gap is said to be an even gap with $o_{j_k} = 0$, and $t_{j_k} = 1$. An odd gap is said to be minimal if $s_{j_k} = 1$, while an even gap is minimal if $s_{j_k} = 2$.

The number of odd gaps between v and H_p is defined as $g_v^o = \sum_{i=1}^{q} o_{j_k}$. Similarly, the number of even gaps is defined as $g_v^e = \sum_{i=1}^{q} t_{j_k}$. Clearly, $\sum_{k=1}^{q} s_{j_k} = p$ and $g_v^e + g_v^o = q$. Note that, for an odd hole, p is odd. So, for $\sum_{k=1}^{q} s_{j_k} = p$ to hold, g_v^o must be odd.

The connection between v and H_p can be classified by (g_v^o, g_v^e, S) where S is an ordered

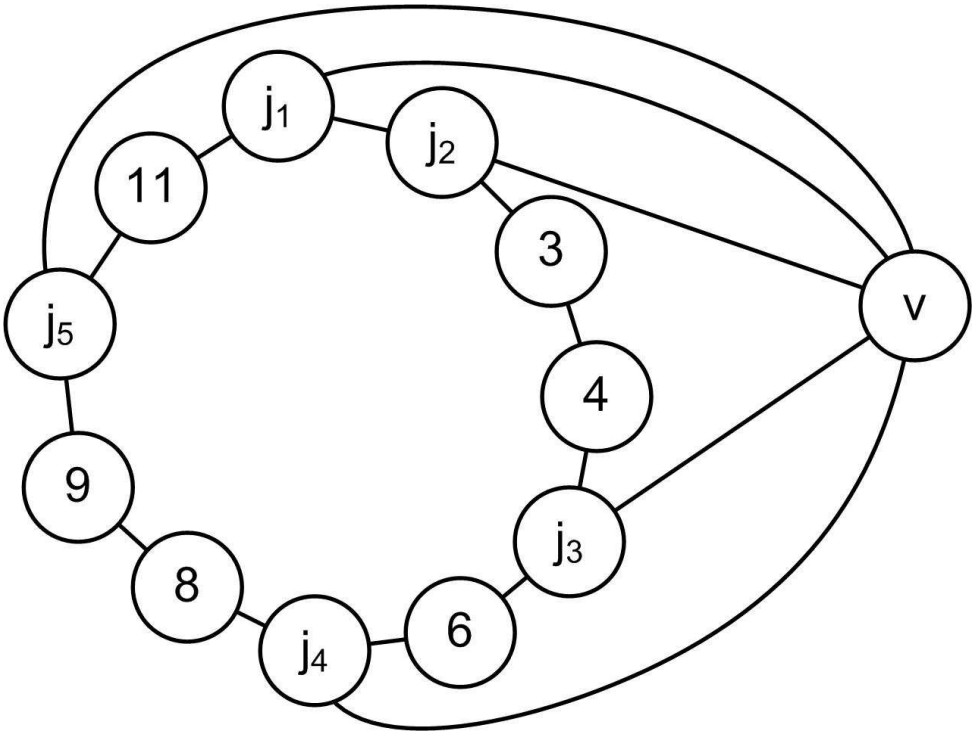

Figure 3.10: A (3,2,(1,3,2,3,2)) connection

set containing the sizes of each gap in the connection, $(s_{j_1}, s_{j_2}, ..., s_{j_q})$. A (g_v^o, g_v^e, S) connection is said to be minimal if $s_{j_k} \in \{1, 2\}$ for all v_{j_k} adjacent to v.

Figure 3.10 provides an example for calculating these values. The value of s_{j_i} can be seen as the number of edges between vertex x_{j_i} and x_{j_i+1}. In this figure, $s_{j_1} = 2-1 = 1$, so the j_1 gap is an odd gap of size 1. Continuing on for all x_{j_i}, $s_{j_2} = 5-2 = 3$, $s_{j_3} = 7-5 = 2$, $s_{j_4} = 10-7 = 3$, and $s_{j_5} = 1-10 = -9 \equiv 2 \pmod{11}$, so $s_{j_5} = 2$. This means there are three odd gaps and two even gaps, so $g_v^o = 3$ and $g_v^e = 2$. The connection to v is denoted $(3, 2, (1, 3, 2, 3, 2))$.

For a graph with induced odd-hole H_p, the inequality $\sum_{i=1}^{p} x_i \leq \lfloor \frac{p}{2} \rfloor$ is valid for the Node Packing polyhedron. The face it induces on P_{NP}^{ch} is of dimension at least $p-1$, as

mentioned in Chapter 2. To lift vertex v into this inequality and increase the inequality's dimension, it is necessary to find the maximum α_v for which the resulting inequality is still valid. The following theorem shows that this coefficient, α_v, can be calculated from the value of g_v^o in a minimal connection.

Theorem 3.2.1 *Given a graph $G = (V, E)$, an induced odd-hole H_p with the associated inequality $\sum_{i \in H_p} x_i \leq \lfloor \frac{p}{2} \rfloor$, and a node v with a minimal (g_v^o, g_v^e, S) connection to H_p, the exact coefficient for lifting v to the odd-hole inequality is $\alpha_v = \lfloor \frac{g_v^o}{2} \rfloor$.*

Proof: To find α_v, it is necessary to know the maximum number of hole nodes that can be in a valid packing that contains v. Alternatively, one needs to find the point P^{NP} with $x_v = 1$ that maximizes $\sum_{i=1}^{p} x_i$. Call this maximum value h_{max}. Clearly α_v is equal to $\lfloor \frac{p}{2} \rfloor - h_{max}$.

Since v is the only node being lifted, it suffices to consider only the induced subgraph of $H_p \cup \{v\}$. Further, since v is in the packing, any node adjacent to v cannot be in the packing thus $v_{j_1}, v_{j_2}, ..., v_{j_q}$ can be removed from the subgraph. After the deletion of these nodes, the resulting subgraph is a collection of vertex disjoint paths.

Since v has a minimal (g_v^o, g_v^e, S) connection to H_p, all that's left in the graph after these deletions is a set of g_v^e disjoint nodes. Clearly, the maximum node packing will contain all of these nodes, so h_{max}=g_v^e. This gives $\alpha_v = \lfloor \frac{p}{2} \rfloor - g_v^e$. Since p is odd, $\alpha_v = \frac{p-1}{2} - g_v^e$. and $2\alpha_v + 1 = p - 2g_v^e$.

From the equations given above, $\sum_{k=1}^{q} s_{j_k} = p$. Since the connection is minimal, there must be g_v^o many s_{j_k}'s equal to 1 and g_v^e many s_{j_k}'s equal to 2. Thus, $g_v^o + 2g_v^e = p$,

and $g_v^o = p - 2g_v^e$. So $2\alpha_v + 1 = g_v^o$, $\alpha_v = \frac{g_v^o - 1}{2}$, and since g_v^o is odd, $\alpha_v = \lfloor \frac{g_v^o}{2} \rfloor$.

□

To extend this result, it is necessary to expand a minimal (g_v^o, g_v^e, S) connection to any general (g_v^o, g_v^e, S) connection. This can be done by increasing the size of the hole through adding pairs of nodes. The new nodes are non-adjacent to v, and are both inserted into the same j_k gap. This way, o_{j_k} and t_{j_k} are not effected. The only change is that and s_{j_k} and p are both increased by 2. It is obvious that this process can be repeated iteratively to create any arbitrary (g_v^o, g_v^e, S) connection. Using this hole expansion method, a stronger theorem can be shown.

Let the hole expanding function $X_i : (\mathbf{Z}^+, \mathbf{Z}^+, \mathbf{Z}_+^{\mathbf{q}}) \rightarrow (\mathbf{Z}^+, \mathbf{Z}^+, \mathbf{Z}_+^{\mathbf{q}})$ be defined as $X_i((g_v^o, g_v^e, S)) = (g_v^o, g_v^e, S')$ where $S' = S + 2\xi_i$ and ξ_i is the i^{th} column of a $q \times q$ identity matrix. This (g_v^o, g_v^e, S') connection describes a new graph, G', with an induced hole H'_{p+2}. The exact lifting coefficient for node v on this new connection is denoted α'_v.

To illustrate this function, consider the example in Figure 3.11. The figure shows the graph described by the connection (3,2,(1,1,2,3,2)), and how the X_3 function effects the associated graph.

Theorem 3.2.2 *Given a graph $G = (V, E)$, an induced odd-hole H_p with the associated inequality $\sum_{i \in H_p} x_i \leq \lfloor \frac{p}{2} \rfloor$, and a node v with a (g_v^o, g_v^e, S) connection to H_p, the exact coefficient for lifting v to the odd-hole inequality is $\alpha_v = \lfloor \frac{g_v^o}{2} \rfloor$.*

Proof: Theorem 3.2.1 shows that, for any minimal (g_v^o, g_v^e, S) connection, $\alpha_v = \lfloor \frac{g_v^o}{2} \rfloor$.

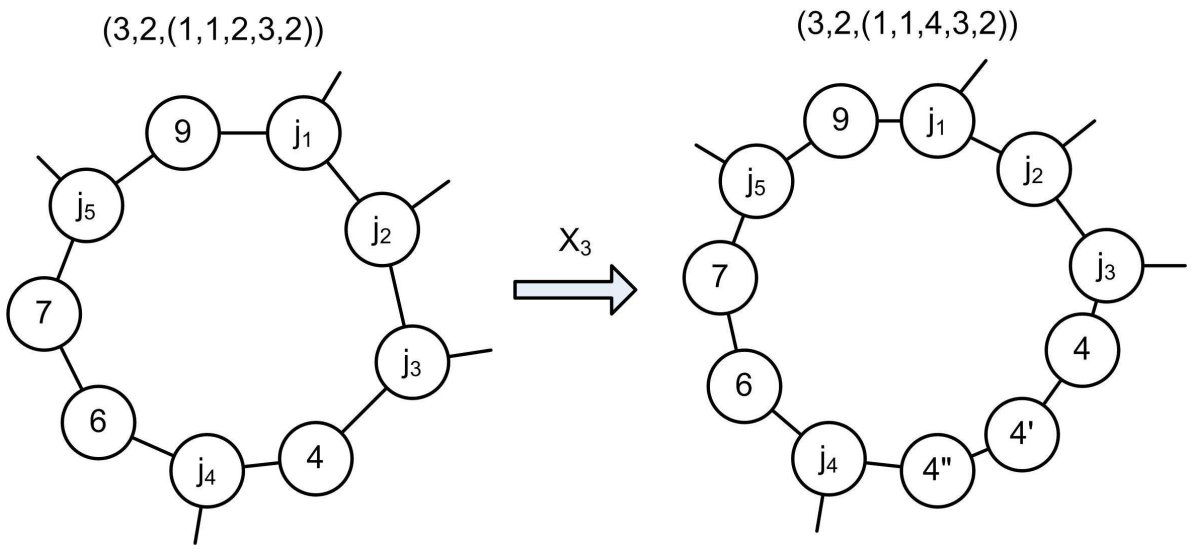

Figure 3.11: The X_i function

Since the X_i function can be applied iteratively to the minimal (g_v^o, g_v^e, S) connection to create any (g_v^o, g_v^e, S) connection, it is sufficient to show that one application of the function does not affect α_v.

Assume that, for some (g_v^o, g_v^e, S) connection, $\alpha_v = \lfloor \frac{p}{2} \rfloor - h_{max} = \lfloor \frac{g_v^o}{2} \rfloor$. Apply X_i to this connection. Then it is necessary to find $\alpha'_v = \lfloor \frac{p+2}{2} \rfloor - h'_{max}$.

As in the last theorem, since v is the only node being lifted, it suffices to consider only the induced subgraph of $H'_{p+2} \cup \{v\}$. Since v is in the packing, the nodes $v_{j_1}, v_{j_2}, ..., v_{j_q}$ cannot be in the packing and can be removed from the subgraph.

After deleting these nodes, the resulting subgraph is a collection of vertex disjoint paths. Further, the only difference between the induced subgraphs on $H'_{p+2} \setminus \{v_{j_1}, v_{j_2}, ..., v_{j_q}\} \cup \{v\}$ and $H_p \setminus \{v_{j_1}, v_{j_2}, ..., v_{j_q}\} \cup \{v\}$ are the two nodes added by the X_i function. Since these two nodes are adjacent, only one of them can be added to the packing. So, $h'_{max} = h_{max} + 1$. This means that $\alpha'_v = \lfloor \frac{p+2}{2} \rfloor - (h_{max} + 1) = \lfloor \frac{p}{2} \rfloor + 1 - h_{max} - 1 =$

α_v. Thus X_i does not affect the lifting coefficient for v and the result follows.

□

The consequence of Theorem 3.2.2 is that finding α_v can be done in $O(n)$ time by checking the adjacencies between v and H_p, then calculating the corresponding g_v^o. This process is referred to as the Odd Gap Lifting (OGL) procedure. This is an improvement over other odd-hole lifting schemes, which typically find these coefficients by setting up small Node Packing problems.

OGL provides a means for creating a sequential lifting scheme for odd-hole inequalities. It provides the exact coefficient for the first node lifted to the inequality, as well as an upper bound on the coefficient for any nodes lifted thereafter. Exact coefficients for the rest of the nodes can be determined by running OGL on groups of nodes.

For some graph $G = (V, E)$ and some $A \subseteq V$, let τ_A be the contraction of A in G. Let the contracted graph be $G' = (V', E')$ where $V' = \{\tau_A\} \cup V \setminus A$ and $E' = \{\{i, j\} : \{i, j\} \in E,\ i, j \notin A\} \cup \{\{\tau_A, j\} : \{i, j\} \in E,\ i \in A,\ j \in V \setminus A\}$. The node τ_A is also known as the supernode of A.

Recall that finding an exact lifting coefficient for node v involves knowing the node packing that maximizes value of the existing inequality, given that v must be included in the packing. Thus, OGL finds the maximum allowable contribution of $\alpha_v x_v$ to the odd-hole inequality, assuming x_v equals one. Similarly, if OGL is run on supernode τ_A, it assumes all member of A are equal to one and finds the maximum allowable contribution of $\alpha_{\tau_A} x_{\tau_A} = \sum_{i \in A} \alpha_i x_i$ to the odd-hole inequality. Clearly, it must be true

that $\alpha_{\tau_A} = \sum_{i \in A} \alpha_i$.

Let $L \subset V \setminus V(H_p)$ be the set of vertices that have been lifted to the odd-hole inequality in some sequential lifting scheme. Call v the next node to be lifted, and let $L_v = \{i : i \in L, \{i, v\} \notin E\}$ be the elements in L not adjacent to v. Let Y be a subset of L_v that defines a valid node packing, so $\{i, j\} \notin E$ for all $i, j \in Y$. Note that Y may be the empty set. If OGL is used to find $\alpha_{\tau_{Y \cup v}}$, then $\alpha_{\tau_{Y \cup v}} - \sum_{i \in Y} \alpha_i$ is a candidate for α_v. Once a candidate has been found for every feasible Y, α_v equals the minimum of these candidates. The smallest candidate value is chosen because it is the one that keeps the inequality valid for all node packings.

To illustrate this, refer back to the sequential lifting example from Figure 2.6. Assume that nodes 8 and 9 have already been lifted to the odd-hole, giving the inequality $\sum_{i=1}^{7} x_i + 2x_8 \leq 3$. The next node to be lifted is node 10. Currently, $L = \{8, 9\}$ and $L_{10} = \{8\}$. There are two subsets of L_{10} that define a valid node packing, namely $\{8\}$ and $\emptyset$. The value of $\alpha_{\tau_{\emptyset \cup \{10\}}}$ is simply the OGL coefficient for node 10, which is 1. For $\tau_{\{8,10\}}$, the contracted graph is shown in Figure 3.12. The OGL coefficient for this supernode is 3, so the candidate for α_{10} is $\alpha_{\tau_{\{8,10\}}} - \alpha_8 = 3 - 2 = 1$. The true value of α_{10} is the minimum of these two candidate values, so $\alpha_{10} = 1$.

Of course, checking all subsets of L_v could necessitate running OGL $2^{|L_v|}$ times. This means that finding lifting coefficients in this manner takes exponential time. Luckily, there are ways to intelligently determine lifting order and cut down on the necessary computation. This is the aim of the Cliques On Odd-holes Lifting procedure, which is

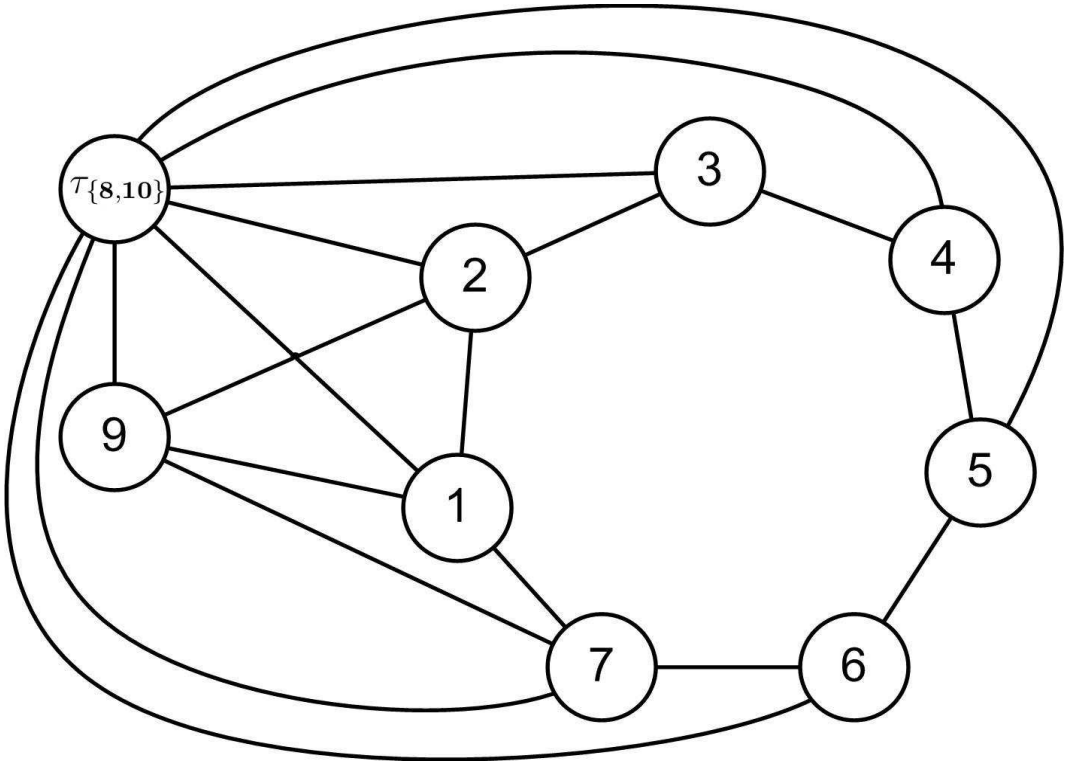

Figure 3.12: Contraction of nodes 8 and 10 from lifting example (Figure 2.6)

discussed in the next section.

3.2.1 OGL Extensions: Cliques On Odd-holes Lifting

To minimize the number of subsets of L_v that need to be considered, another well known graph structure can be utilized. Any node in a clique is connected to all other clique members. So if the only nodes lifted before node v are members of a clique containing v, node v may be lifted in with its OGL coefficient. Then when determining the lifting coefficient of a node outside of the clique, it is immediately known that any subset of L_v that contains two members of the clique does not constitute a valid node packing. Thus, by lifting large cliques of nodes together, the number of subsets of L_v that need

to be checked is cut down drastically. This is the idea behind the Cliques On Odd-holes Lifting (COOL) procedure.

The COOL procedure seeks to efficiently determine a facet-defining inequality based on an odd-hole. Of course, depending on lifting order, an odd hole can give rise to multiple facet-defining inequalities. COOL seeks an order that lowers the necessary computational load. The main idea is to keep the number of feasible subsets of L_v as low possible.

The first step in the COOL procedure is to determine the OGL coefficient for each node in $V \setminus H_p$. This only requires $O(n^2)$ effort, and the results are valuable. Since OGL gives an upper bound for the sequential lifting coefficient, it is immediately known that any node with an OGL coefficient of zero will be lifted in with that zero no matter when it is lifted. Thus, to keep $|L_v|$ as low as possible, these nodes are removed from the graph.

Next, the COOL procedure examines the nodes whose OGL coefficient is at the maximum possible value, $\lfloor \frac{p}{2} \rfloor$. These are nodes that are adjacent to every node in H_p, and thus $H_p \cup \{v\}$ induces a wheel. If a node is lifted in with this value, any node not adjacent to it must be lifted in with a zero. Again, zero coefficients mean that a node can be removed from the graph. It is thus advantageous to lift in these nodes whose OGL coefficient is $\lfloor \frac{p}{2} \rfloor$.

Of course, the natural inclination is to add as many of these nodes as possible. So, ideally, the next COOL step would be to find the maximum clique containing these

nodes. Unfortunately, finding a maximum clique is $\mathcal{NP}$-Hard [32]. There are, however, many polynomial time algorithms for finding a maximal clique. Once COOL finds a maximal clique of these wheel-inducing nodes, the identity of each member is stored. Any node not adjacent to every clique member will be lifted in with a zero coefficient and can be removed from the graph.

After this is done, COOL proceeds by finding large cliques of unlifted nodes. Every member of the first clique may be lifted with its OGL coefficient. Beyond the first clique, the lifting coefficient must be found by running OGL on the supernodes created by v and each feasible subset of L_v. Since COOL stores the identity of each clique lifted, the structure of all such feasible subsets is known.

Let K^i denote the i^{th} clique lifted in the COOL procedure. A feasible subset $Y \subseteq L_v$ is of the form $\{y_1, y_2, ..., y_k\}$ where k is the number of cliques lifted so far, and $y_i \in K^i$ or $y_i = \emptyset$. The size of any one clique is bounded by n, the number of nodes. Clearly, the number of feasible subsets is then bounded by n^k.

As COOL is running OGL for each feasible subset of L_v, if zero is ever found as a candidate for α_v, node v is removed from the graph. If a clique ever loses one of it's members, COOL checks all other unlifted nodes to determine if the clique is still maximal. If it is no longer maximal, another node is added to the clique.

Once all nodes are either lifted or known to be zero, the procedure terminates and outputs a COOL inequality. This inequality is facet defining and may be added to the IP formulation of the Node Packing problem. The above-described method was coded

in C++ for a computational study. Methodology and results of this study are found in chapter 4.

3.2.2 A COOL Example

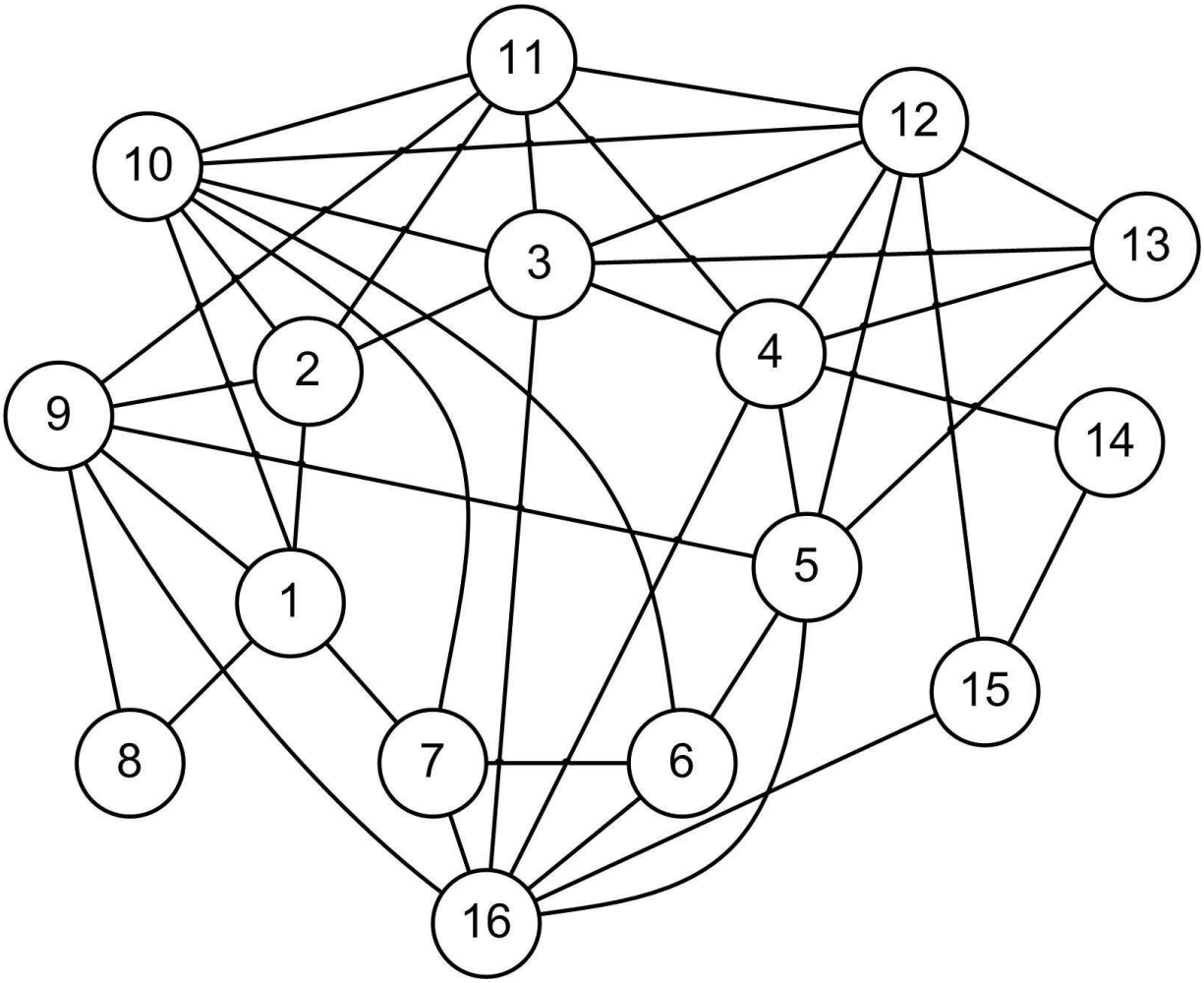

Figure 3.13: COOL example graph

Consider the graph shown in Figure 3.13. Notice the H_7 induced by nodes 1-7 with associated inequality $\sum_{i=1}^{7} x_i \leq 3$. The COOL procedure initiates by finding the OGL coefficient for nodes 8-16. The results are shown in Table 3.1. Since nodes 8, 14, and

15, have an OGL coefficient of 0, they can be removed from further consideration.

Table 3.1: Initial OGL coefficients for COOL example

Node	8	9	10	11	12	13	14	15	16
OGL Coefficient	0	1	2	1	1	1	0	0	2

Next, COOL searches for nodes whose OGL coefficient equals $\lfloor \frac{p}{2} \rfloor$, which in this case is 3. None exist, so the procedure proceeds by finding a maximal clique of the remaining nodes. Say that it finds the K_3 induced by nodes 10, 11, and 12. Each of these nodes is lifted in with its OGL coefficient, making the inequality $\sum_{i=1}^{7} x_i + 2x_{10} + x_{11} + x_{12} \leq 3$. Now, $L = K^1 = \{10, 11, 12\}$, and COOL looks for another maximal clique.

Say that the clique detection procedure finds nodes 9 and 16. To lift in node 9, the subsets of L_9 must be known. In this case, $L_9 = \{10, 12\}$ and the supernodes that must be checked are $\tau_{\{9,10\}}, \tau_{\{9,12\}}$, and $\tau_{\{9\}}$. The value of $\alpha_{\tau_{\{9\}}}$ is already known to be 1. Then $\alpha_{\tau_{\{9,10\}}}$ is calculated to be 2, so the candidate for α_9 is $\alpha_{\tau_{\{9,10\}}} - \alpha_{10} = 2 - 2 = 0$. Since a candidate was calculated to be 0, node 9 is removed from the graph.

Moving on to the next node in the clique, $L_{16} = \{10, 11, 12\}$. The candidate values from $\tau_{\{10,16\}}, \tau_{\{11,16\}}, \tau_{\{12,16\}}$, and $\tau_{\{16\}}$ are found to be 1, 1, 1, and 2 respectively. The minimum of these candidates is 1, so $\alpha_{16} = 1$.

The second clique has now been lifted, but since it lost one of its members (node 9), COOL looks at the remaining unlifted nodes to determine if any can be added to the clique. The only candidate is node 13, which is not adjacent to 16, so it cannot be added to the clique. This means that the second clique is now done, $K^2 = \{16\}$,

x_1	1	0	0	1	0	1	0	0	1	0	0	0	1	0	0	0
x_2	0	1	0	0	1	0	1	1	0	1	0	0	0	0	0	0
x_3	1	0	1	0	0	1	0	0	1	0	0	0	0	0	0	0
x_4	0	1	0	1	0	0	1	1	0	1	1	0	0	0	0	0
x_5	1	0	1	0	1	0	0	0	1	0	0	1	0	0	0	0
x_6	0	1	0	1	0	1	0	1	0	0	0	0	1	0	0	0
x_7	0	0	1	0	1	0	1	0	0	1	0	1	0	0	0	0
x_8	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
x_9	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
x_{10}	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	1
x_{11}	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
x_{12}	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0
x_{13}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
x_{14}	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
x_{15}	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
x_{16}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

Figure 3.14: Affinely independent points for COOL example

$L = \{10, 11, 12, 16\}$, and the inequality is now $\sum_{i=1}^{7} x_i + 2x_{10} + x_{11} + x_{12} + x_{16} \leq 3$.

COOL proceeds by finding the next maximal clique. In this case, node 13 is the only node left, so it is the final clique. The subsets of $L_{13} = \{10, 11, 16\}$ that need to be checked are the ones that contain at most one member of K^1 and one member of K^2. The sets that satisfy this condition are $\emptyset, \{10\}, \{11\}, \{16\}, \{10, 16\}$ and $\{11, 16\}$. While checking these candidates, COOL finds that $\alpha_{\tau_{\{10,13,16\}}} = 3$, making the candidate for α_{13} equal to $\alpha_{\tau_{\{10,13,16\}}} - \alpha_{10} - \alpha_{16} = 3 - 2 - 1 = 0$. So node 13 is removed from the graph, and the procedure is complete. The final COOL inequality is $\sum_{i=1}^{7} x_i + 2x_{10} + x_{11} + x_{12} + x_{16} \leq 3$, which is facet defining in P_{NP}^{ch} as shown by the points in Figure 3.14.

Chapter 4

Computational Results

Because OGL does not involve solving an integer program to find lifting coefficients, it stands to reason that COOL would be more efficient than traditional methods for lifting to an odd-hole inequality. This hypothesis was tested on various random problems, using C++ coding language and the ILOG CPLEX 10.0 callable library [13]. The study was performed on an Intel (R) Core i7 computer with a 1.58 GHz processor and 3.0 GB of RAM.

The main goal of this study was to determine the time necessary to create a facet defining inequality from an odd-hole inequality through sequential lifting. This was done using two methods; first by COOL, then again by solving Node Packing problems for each variable to be lifted.

The study was executed on randomly generated weighted Node Packing instances. Objective coefficient values are uniformly distributed between 1000 and 2000. Node

adjacencies in each problem were determined based on some edge probability parameter. This parameter is a number between zero and one that represents the probability that any two nodes are adjacent. Once the graph is created, the code uses a depth-first search method to find 20 odd-holes of a specified size. Each hole is lifted by both methods to create facet defining cuts.

The COOL procedure is run as outlined in the previous chapter, using a well-known method for finding maximal cliques: After identifying which nodes are eligible to be lifted, the induced subgraph on this set of nodes is considered. The degree of each node is calculated, and the node with the minimum degree is removed from the subgraph. Nodes are removed until all d remaining nodes have degree $d-1$. Ties are broken first by the lowest objective function coefficient, then by the lowest variable index.

The second procedure creates a Node Packing problem to determine exact lifting coefficients for every variable, after completing two preprocessing steps. The first step identifies all nodes with two or fewer adjacencies in the hole. It has been observed previously that such nodes will never have a positive lifting coefficient in the odd-hole inequality [39], a result which now follows directly from the theorems provided in section 3.2.

The second preprocessing step applies an idea from COOL: It searches for nodes adjacent to every node in the hole (nodes that, together with the hole, induce a wheel) and finds a maximal clique of these nodes. These nodes are lifted with a coefficient of $\lfloor \frac{p}{2} \rfloor$. As in COOL, any node not adjacent to every member of this clique is lifted with a

zero coefficient.

After this preprocessing, the procedure (hereafter referred to as the NP procedure) creates a Node Packing problem to find the exact lifting coefficient of every unlifted node. This problem is solved using the default settings of CPLEX's mixed integer programming solver. Lifting is done in ascending order of variable indices.

Both COOL and the NP procedure lift to each of the 20 holes found by the depth-first search. The total lifting time is tracked for each procedure. Additionally, as a measure of cut strength, the LP relaxation is solved for the original problem, the problem with COOL cuts, and the problem with NP cuts. All LPs are solved with CPLEX's linear programming optimizer. The change in objective value gives a rough measure of the usefulness of each procedure's cuts. The study also keeps track of the number of variables with non-zero coefficients in each inequality, which helps to indicate the number of variables that will be affected by the cut.

Three input variables control what type of problem is solved. These variables are the size of the graph, the edge probability parameter, and the size of hole to be lifted. Prior research [39] has suggested that odd-hole inequalities are most likely to be useful on graphs with lower edge densities. For this reason, the study was run with edge probability parameters of 0.1, 0.2, and 0.3. The number of nodes was either 100, 200, or 300. The hole sizes used were 5, 9, and 13. This gives 27 total combinations, each of which was run ten times.

Statistics for each factor combination are provided in Table 4.1. This table contains

the average of each value over all ten replications. The "Cut Time" columns display the average time taken to lift all 20 hole inequalities. Entries in the "IP/COOL" column show IP cut time divided by COOL cut time. The "LP % Change" columns contain the average percentage decrease in the problem's linear relaxation solution when cuts were added. The "# Vars" columns contain the average number of variables in the inequalities with non-zero coefficients, including the hole nodes.

The data suggests that COOL is indeed a significant improvement over NP in most instances. Over all trials, the NP procedure took 10% longer than COOL to complete the cuts. This result is heavily skewed to the 0.3 edge density, size 13 hole problems that took longest to run. In the worst case studied, COOL was slower than NP by a factor of three. However, in most instances, COOL was hundreds to thousands of times faster than NP.

On the lowest density graphs, COOL was uniformly fast no matter the value of the other parameters. This is encouraging, because low density graphs are where odd-hole inequalities are most useful. However, COOL's limitations were seen on the most edge-dense graphs when the initial hole sizes are large. In particular, the only times that NP outperformed COOL were on the 0.3 density, size 13 hole instances with 200 and 300 nodes. In both of these situations, COOL displayed large variations in total cut times.

In the 200 node case, COOL was actually faster than NP in seven of the ten replications. However, COOL's total cut times ranged from 21 seconds to 197 seconds, while NP was always between 61 and 70 seconds. Over all ten replications, standard deviations

Table 4.1: COOL vs. NP computational results

Edge Density	Hole Size	# of Nodes	COOL Cut Time (Sec)	NP Cut Time (Sec)	NP/ COOL	COOL LP % Change	NP LP % Change	# Vars in COOL Cuts	# Vars in NP Cuts
0.1	5	100	0.000	0.091	-	10.0	10.1	5.3	5.3
		200	0.005	0.422	91.67	8.1	8.2	5.6	5.6
		300	0.005	0.972	211.28	7.3	7.2	6.0	6.0
	9	100	0.002	0.666	416.06	11.9	12.2	10.4	10.4
		200	0.005	3.058	650.60	11.1	11.1	11.3	11.3
		300	0.003	6.224	2007.61	11.1	10.6	12.1	12.0
	13	100	0.002	1.911	1273.93	14.1	13.9	15.8	15.8
		200	0.011	7.774	694.06	14.5	14.4	17.6	17.5
		300	0.014	14.66	1032.47	13.7	13.2	18.5	18.3
0.1 Density Averages			**0.01**	**3.98**	**786.31**	**11.3**	**11.2**	**11.4**	**11.3**
0.2	5	100	0.002	1.216	759.88	20.4	19.8	6.9	6.8
		200	0.011	4.453	404.85	17.5	17.0	7.8	7.6
		300	0.006	9.105	1468.52	15.6	13.9	8.4	8.0
	9	100	0.006	5.267	849.52	26.4	24.7	14.2	13.8
		200	0.030	23.27	780.84	24.8	21.9	16.1	15.4
		300	0.084	40.87	485.38	22.5	19.3	17.1	16.1
	13	100	0.033	9.394	283.79	28.2	26.9	21.8	21.1
		200	0.414	49.42	119.37	29.2	24.9	25.0	23.7
		300	1.581	78.19	49.44	27.1	23.1	26.4	25.1
0.2 Density Averages			**0.24**	**24.58**	**102.0**	**23.5**	**21.3**	**16.0**	**15.3**
0.3	5	100	0.002	3.480	2174.94	30.2	28.2	8.7	8.5
		200	0.013	15.47	1227.80	25.9	23.8	9.7	9.7
		300	0.020	36.55	1818.33	22.0	20.4	10.2	10.3
	9	100	0.034	12.47	363.52	36.5	34.1	18.6	17.7
		200	0.408	50.71	124.28	34.7	30.0	21.4	20.2
		300	1.802	120.59	66.94	32.6	25.8	23.0	21.2
	13	100	1.116	21.60	19.36	40.6	37.7	29.2	27.6
		200	74.92	64.77	0.86	39.0	33.6	34.3	31.6
		300	618.32	184.05	0.30	36.2	30.7	36.4	34.1
0.3 Density Averages			**77.40**	**56.63**	**0.73**	**33.1**	**29.4**	**21.3**	**20.1**
Overall Averages			**25.88**	**28.39**	**1.10**	**22.6**	**20.6**	**16.2**	**15.6**

were found to be 53.1 for COOL, while only 2.93 for NP. A two sample t-test found no significant difference in mean cutting time with a p-value of 0.561. The 95% confidence interval for the difference of the two means (COOL - NP) is (-27.9, 48.2).

In the 300 variable case, NP was faster than COOL in every instance. COOL took anywhere from 227 to 1717 seconds, with a standard deviation of 442. NP was again more stable, with total times ranging from 179 to 192 and a standard deviation of 4.43. A two sample t-test found a statistically significant difference in the means with a p-value of 0.013. The 95% confidence interval for the difference of the two means (COOL - NP) is (118, 750).

In general, COOL's large time deviations only existed in the high-density, large hole cases. The basic problem occurs because multiple vertices with non-zero coefficients exist, and thus COOL's exponential run time manifests. Consequently, COOL should probably not be used in these type of instances.

An interesting result is that the COOL inequalities tended to contain more variables with non-zero coefficients. This may be due to the fact that COOL lifts entire cliques at one time, instead of looking for new nodes individually. The COOL cuts also reduced the LP relaxation solution by a higher percentage, suggesting that they are just as strong or stronger than the randomly lifted NP cuts. This is likely due to the higher number of non-zero coefficients, and the clique tie-breaker rule that eliminates nodes with low objective value coefficients.

Since the results of this study suggested that COOL runs efficiently on the lowest

Table 4.2: Additional computational results for edge density 0.1

Problem Size	Hole Size	COOL Cut Time (Sec)	NP Cut Time (Sec)	NP/ COOL	COOL LP % Change	NP LP % Change	# Vars in COOL Cuts	# Vars in NP Cuts
1500	13	1.92	1147.94	599.20	8.7	7.4	22.4	21.5
1500	17	19.98	2076.65	103.96	10.6	8.7	30.0	28.9
1500	21	209.52	3122.49	14.90	12.0	9.5	37.4	36.2
2000	5	0.10	174.21	1684.84	2.9	2.6	7.8	7.5
3000	5	0.25	628.55	2514.19	2.4	2.0	8.4	7.8
4000	5	0.45	1548.44	3464.07	1.8	1.5	8.5	7.9
AVERAGE		38.70	1449.71	37.46	5.0	4.1	19.1	18.3

density graphs, more tests were run with an edge density parameter of 0.1. These tests were run with greater initial hole sizes and with a larger number of variables. Five replications of each scenario were run. Results of these trials are shown in Table 4.2.

This data suggests that COOL continues to perform well with a large number of variables as long as the initial hole is small. In these cases, COOL completed its cuts thousands of times faster than NP. As hole sizes grow, the gap between NP and COOL decreases noticeably.

These extra trials continue to show that COOL cuts decrease the LP relaxation objective value more than the NP cuts. On average, COOL dropped the objective value 1% more than NP. COOL also continues to generate inequalities with more non-zero coefficients.

Looking at all of the data, it can be said that COOL performs well on graphs with a low edge density. COOL is also very efficient when lifting to a small initial hole. COOL struggles to compete with NP on medium density graphs when lifting to larger sized

holes.

These results suggest that the creation of a modified COOL algorithm could be beneficial. This algorithm would run COOL in the right circumstances, then switch to NP or some approximate method when appropriate. For instance, it appears that inequalities with more than 30 non-zero coefficients cause problems for COOL. A modified COOL algorithm may decide to approximate lifting coefficients after the first 30 nodes have been lifted. Such a procedure could still create useful cuts, while minimizing computational time.

Chapter 5

Conclusion and Future Research

This research proposed two different methods for lifting in the Node Packing problem. The first was the Simultaneous Lifting Expansion, which combines two known inequalities to create one stronger inequality. Simultaneous lifting in Node Packing problems has only been addressed once in the literature [14], so this research represents a significant advancement to this area. SLE allows for the creation of valid inequalities that are unachievable by currently known methods.

The second contribution is the Cliques On Odd-holes Lifting procedure. This method takes an odd-hole inequality and sequentially uplifts variables to create a facet defining cut. This method is unique in that it does not solve an Integer Program to find lifting coefficients. Instead, it makes use of the Odd Gap Lifting procedure, which determines lifting coefficients by examining node adjacencies.

Several Node Packing branch and cut algorithms have made use of lifted odd-hole

inequalities [29, 38, 39, 49]. Research has suggested that these inequalities are generally not as useful as other cuts (e.g. clique inequalities), but are more likely to be useful under certain circumstances. In particular, lifted odd-hole inequalities are most useful when edge densities are low. This is exactly the situation in which COOL performs best, suggesting that COOL is a viable option for these branch and cut procedures. As such, an implementation of COOL in the right circumstances may be beneficial in solving IPs.

5.1 Future Research

The work done for this thesis points to several areas for future research. Possible future research exists for both theory and computation. In particular, SLE may motivate more theoretical research, while COOL encourages more computational research.

While SLE provides interesting theoretical properties, the requisite underlying structures are unlikely to occur in general Node Packing instances. As such, application of SLE would be of little practical use. Still, there is little known about simultaneous lifting in Node Packing problems, and continued research in this area may yet provide practical results and new facet defining structures.

Concepts from SLE may also be applied to simultaneous lifting procedures in other classes of Integer Programs. SLE takes two facet defining structures with a common node, and creates a new facet defining cut through a mix of up and down lifting. This is, to the best of the author's knowledge, the first simultaneous lifting method that can do this. It may be possible to create similar methods in other Integer Program-

ming problems. For example, there may be some analogous method for combining two overlapping cover inequalities in a knapsack problem.

Computational results for COOL indicate that it may be fit for implementation in commercial software under the right conditions. More computational studies should be completed before this can be known for sure. Further research may point to particular conditions when lifted odd-hole inequalities are beneficial in solving IPs.

While COOL provides an improved method for lifting to odd-holes, other researchers have suggested that approximated inequalities can be found quickly and perform about as well as exact inequalities [29, 38]. It may be useful to complete a computational study comparing run-times and cut qualities of COOL with an approximate method. Such a study could highlight the advantages of using COOL over an approximate method.

In every size problem tested for this thesis, COOL performed admirably in low edge weight graphs. However, the procedure's exponential nature began to show on medium density problems when lifting to larger holes. It would be beneficial to further define COOL's limitations to determine when approximation methods are preferable to COOL. This data would inform the creation of a modified COOL procedure, which runs COOL in situations when it is most advantageous, and uses another method otherwise.

Finally, once a modified COOL procedure has been created, it could be implemented in a branch and cut routine and tested against a commercial optimization software. This could further define the conditions in which lifted odd-hole inequalities improve IP solving time. Similar tests have been run previously, but not with the COOL method.

Bibliography

[1] Arunapuram, S., K. Mathur, Solow, D. (2003). "Vehicle Routing and Scheduling with Full Truckloads," *Transportation Science*, **37** (2), 170-82.

[2] Atamtürk, A., Nemhauser, G., Savelsbergh, M. (2000). "Conflict Graphs in Solving Integer Programming Problems," *European Journal of Operational Research*, **121**, 40-55.

[3] Atamtürk, A. (2004). "Sequence Independent Lifting for Mixed-Integer Programming," *Operations Research*, **52** (3), 487-491.

[4] Assuncao, T., Furtado, V. (2008). "A Heuristic Method for Balanced Graph Partitioning: an Application for the Demarcation of Preventive Police Patrol Areas," *Advanced in Artificial Intelligence - IBERAMIA 2008. Proceedings 11th Ibero-American Conference on AI*, 62-72.

[5] Avella, P., Boccia, M., Sforza, A. (2004). "Resource Constrained Shortest Path Problems in Path Planning for Fleet Management," *Journal of Mathematical Modelling and Algorithms*, **3** (1), 1-17.

[6] Balas, E., Zemel, E. (1984). "Lifting and Complementing Yields All the Facets of Positive Zero-one Programming Polytopes," in *Mathematical Programming, Proceedings of the International Conference on Mathematical Programming*, R.W. Cottle et al., eds., 13-24.

[7] Balas, E., Ng, S. (1989). "On the Set Covering Polytope. II, Lifting the Facets with Coefficients in 0,1,2," *Mathematical Programming*, **45**, 1-20.

[8] Balas, E. (1999). "New Classes of Efficiently Solvable Generalized Traveling Salesman Problems," *Annals of Operations Research*, **86**, 529-88.

[9] Biggs, N., Lloyd E., Wilson, R. (1986). *Graph Theory, 1736-1936*. Oxford University Press.

[10] Bolton, J. (2009). "Synchronized Simultaneous Lifting in Binary Knapsack Polyhedra," *MS Thesis*, Department of Industrial and Manufacturing Systems Engineering, Kansas State University.

[11] Bourgeois, N., Della Croce, F., Escoffier, B., Murat, C., Paschos, V. (2009). "Probabilistic Graph-coloring in Bipartite and Split Graphs," *Journal of Combinatorial Optimization*, **17** (3), 274-311.

[12] Carmia, M., Dell'Olmo, P., (2002). "Constraint Propagation in Graph Coloring," *Journal of Heuristics*, **8** (1), 83-107.

[13] The CPLEX Solver on ILOG's Home Page, http://www.ilog.com/.

[14] Conley, C. (2009). "Cliqed Holes and Other Graphic Structures for the Node Packing Polytope," *MS Thesis*, Department of Industrial and Manufacturing Systems Engineering, Kansas State University.

[15] Dell'Olmo, P., Speranza, M. (1996). "Graph Models for Multiprocessor Scheduling Problems with Precedence Constraints," *Foundations of Computing and Decision Sciences*, **21**, 17-29.

[16] Easton, K., Nemhauser, G., Trick, M. (2003). "Solving the traveling tournament problem: A combined integer programming and constraint programming approach," *Practice and Theory of Automated Timetabling IV. 4th International Conference, PATAT 2002*, Selected Revised Papers (*Lecture Notes in Comput. Sci.* Vol.2740), 2003, p 100-9.

[17] Easton, T., Hooker, K. "Simultaneously Lifting Sets of Binary Variables into Cover Inequalities for Knapsack Polytopes," *Discrete Optimization, Special Issue: In Memory of George B. Dantzig*, **5** (2) May 2008, 254-261.

[18] Erlebach, T., Hall, A. (2004). "NP-Hardness of Broadcast Scheduling and Inapproximability of Single-Source Unsplittable Min-Cost Flow" *Journal of Scheduling*, **7** (3), 223-41.

[19] Finn, Frank J. (1973). "Integer Programming, Linear Programming and Capital Budgeting." *Abacus*, **13**, 180-192.

[20] Frangioni, A., Gentile, C. (2007). "Prim-based Support-graph Preconditioners for Min-cost Flow Problems," *Computational Optimization and Applications*, **36** (2), 271-287.

[21] Goczyla, K., Cielatkowski, J. (1995). "Optimal Routing in a Transportaion Network," *European Journal of Operational Research*, **87**, 214-22.

[22] Gomory, R. (1969). "Some Polyhedra Related to Combinatorial Problems," *Linear Algebra and its Applications*, **2**, 451-558.

[23] Grabowski, J., Skubalska, E. (1985). "Optimization of the Transportation Network Structure with Respest to the Flow Costs," *Archiwum Automatyki i Telemechanika*, **30**, 3-21.

[24] Gu, Z., Nemhauser, G., Savelsbergh, M. (2000). "Sequence Independent Lifting in Mixed Integer Programming," *Journal of Combinatorial Optimization*, **4**, 109-129.

[25] Guerriero, G., Tseng, P. (2002). "Implementation and Test of Auction Methods for Solving Generalized Network Flow Problems with Separable Convex Cost," *Journal of Optimization Theory and Applications*, **115** (1), 113-44.

[26] Gutin, G., Karapetyan, D. (2010). "A Memetic Algorithm for the Generalized Traveling Salesman Problem," *Natural Computing*, **9** (1), 47-60.

[27] Harely, E., Bonner, A., Goodman, N. (2001). "Uniform Integration of Genome Mapping Data Using Intersection Graphs," *Bioinformatics*, **17** (6), 487-94.

[28] Harris, A. (2010). "Generating an Original Cutting-plane Algorithm in Three Sets (GO CATS)," *MS Thesis*, Department of Industrial and Manufacturing Systems Engineering, Kansas State University.

[29] Hoffman, K., Padberg, M. (1993). "Solving Airline Crew Scheduling Problems by Branch-and-Cut," *Management Science*, **39** (6), 657-82.

[30] Irani, S., Leung, V. (2003). "Scheduling with Conflicts on Bipartite and Interval Graphs," *Journal of Scheduling*, **6**, 287-307.

[31] Iwamura, K., Liu, B. (1999). "Dependent-Chance Integer Programming Applied to Capital Budgeting." *Journal of the Operations Research Society of Japan*, **11**, 117-127.

[32] Karp, R. (1972). "Reducibility among Combinatorial Problems," in *Complexity of Computer Computations*, R. E. Miller and J. W. Thatcher, eds., Plenum Press, New York 85-103.

[33] Kaufman, D., Nonis, J., Smith, R. (1998). "A Mixed Integer Linear Programming Model for Dynamic Route Guidance," *Transportation Research, Part B (Methodological)*, **32B** (6), Aug. 1998, p 431-40.

[34] Kubik, L., (2009) "Simultaneously Lifting Multiple Sets in Binary Knapsack Integer Programs," *MS Thesis*, Department of Industrial and Manufacturing Systems Engineering, Kansas State University.

[35] Kumar, S., Gupta, P. (2003). "An Incremental Algorithm for the Maximum Flow Problem," *Journal of Mathematical Modelling and Algorithms*, **2** (1), 1-16.

[36] Lam, M., Liu, Y. (2006). "Active Sensor Network Deployment and Coverage Enhancement Using Circle Packings," *2006 IEEE International Conference on Robotics and Biomimetics, ROBIO 2006*, 520-525.

[37] Liberti, L., Alfandari, L., Plateau, M. (2009). "Edge Cover by Connected Bipartite Subgraphs," *Annals of Operations Research*, **167**, 1-23.

[38] Mannino, C., Sassano, A. (1994). "An Exact Algorithm for the Maximum Stable Set Problem," *Computational Optimization and Applications*, **3** (3), 243-58.

[39] Nemhauser, G., Sigismondi, G. (1992). "A Strong Cutting Plane/Branch-and-Bound Algorithm for Node Packing," *The Journal of the Operational Research Society*, **43** (5), 443-457.

[40] Olmes, Z., Kun Myon Choi, Min Young Chung, Tae-Jin Lee, Hyunseung Choo (2005). "RWA Based on Approximated Path Conflict Graphs in Optical Networks," *Computational Science and Its Applications - ICCSA*, 448-58.

[41] Padberg, M. (1973). "On the Facial Structure of Set Packing Polyhedra," *Mathematical Programming*, **5**, 199-215.

[42] Pape, U. (1974). "Implementation and Efficiency of Moore-algorithms for the Shortest Route Problem," *Mathematical Programming*, **37** (1), 212-22.

[43] Park, K. (1997). "Lifting Cover Inequalities for the Precedence-Constrained Knapsack Problem," *Discrete Applied Mathematics*, **72** (3), 219-241.

[44] Plesnik, J. (2001). "Minimum Cost Edge Subset Covering Exactly k Vertices of a Graph," *Journal of Combinatorial Optimization*, **5** (3), 275-86.

[45] Raj, V. (2008). "Better Performance of Neural Networks Using Functional Graph for Weather Forecasting," *12th WSEAS International Conference of Computers,* 826-31.

[46] Ruiz, R., Maroto, C., Alcaraz, J. (2004). "A Decision Support System for a Real Vehicle Routing Problem," *European Journal of Operational Research*, **153** (3), 16 March 2004, 593-606.

[47] Toth, P. (1997). "An Exact Algorithm for the Vehicle Routing Problem with Backhauls," *Transportation Science*, **31** (4), Nov. 1997, 372-85.

[48] Urban, T. (2003). "Scheduling Sports Competitions on Multiple Venues," *European Journal of Operational Research*, **148** (2), 16 July 2003, 302-11.

[49] Verweij, B., Aardal, K. (1999). "An Optimisation Algorithm for Maximum Independent Set with Applications in Map Labelling," *Algorithms - ESA* 696. Heidelberg: Springer Berlin.

[50] Wang Qiang, Gao Bin, Jia Cui-xia (2008) "Complete Graph Algorithm Based on Four Level Forecast Model," *Journal of China Academy of Electronics and Information Technology*, **3**, 623-6.

[51] Waterer, H., Johnson, E., Nobili, P., Savelsbergh, M. (2002) "The Relation of Time Indexed Formulations of Single machine Scheduling Problems to the Node Packing Problem," *Mathematical Programming, Series B*, **93**, 477-494.

[52] Yun-Wu Huang, Ning Jing, Rundensteiner, E.A. (2000). "Optimizing Path Query Performance: Graph Clustering Strategies," *Transportation Research Part C (Emerging Technologies)* **8C**, 381-408.

[53] Zwaneveld, P., Kroon, L., van Hoesel, S. (2001). "Routing Trains Through a Railway Station Based on a Node Packing Model," *European Journal of Operational Research*, **128**, 14-33.