A NEW HIERARCHY OF RELAXATIONS FOR 0-1 MIXED INTEGER PROBLEMS WITH APPLICATION TO SOME SPECIALLY STRUCTURED PROBLEMS

by

Patrick J. Driscoll

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of
DOCTOR OF PHILOSOPHY
in
Industrial and Systems Engineering

APPROVED:

H. D. Sherali, Chairman

J. A. Nachlas

L. T. Watson

C. P. Koelling

S. C. Sarin

April, 1995

Blacksburg, Virginia

Abstract

A new hierarchy of relaxations is developed that extends the Reformulation-Linearization Technique (RLT) of Sherali and Adams (1989, 1990). This hierarchy, referred to as (**RLT1**), provides a unifying framework for constructing a spectrum of continuous relaxations spanning from the linear programming relaxation to the convex hull representation for linear mixed integer 0–1 problems, and is particularly designed to exploit explicit or implicit special structures defined by the constraints of a problem. Specifically, inherent special structures are exploited by identifying specific classes of multiplicative factors that can be applied to the original mathematical formulation of a problem to reformulate it as an equivalent polynomial programming problem. Subsequently, this resulting problem is linearized to produce a tighter relaxation in a higher dimensional space. This general framework permits one to generate a hierarchical sequence of tighter relaxations leading up to the convex hull representation. Several classes of constraints are presented to demonstrate how underlying special structures, including generalized upper bounding (GUB), variable upper bounding (VUB), covering, partitioning and packing constraints, as well as sparsity, can be exploited within this framework. For some types of structures, low level relaxations are exhibited to recover the convex hull of integer feasible solutions. An alternative partial application of this new hierarchy is also presented, along with a discussion of some additional cases that might lend themselves to such a scheme. Additional ideas for further strengthening **RLT1**–based constraints by using conditional logical implications, as well as relationships with sequential lifting, are also presented.

This new **RLT1** is applied in detail to the set packing problem and several formulations of the Asymmetric Traveling Salesman Problem (ATSP). Computational experimentation is performed to illustrate the relative strength of **RLT1** relaxations in comparison to those obtained using other methods. A new class of valid inequalities for the 3-index TSP is also presented. Finally, the dissertation concludes with comments concerning extensions and parallel implementations of **RLT1**.

Dedication

This work is dedicated to several important people without whom this dissertation would not be in existance. It is dedicated to the members of my committee, who have all contributed immensely, each in their own way, to furthering my education, refining my understanding of academia, and expanding my professional horizons. It is dedicated to Hanif D. Sherali, the chairman of my committee, and my academic mentor. I have learned much more than this work is capable of communicating. It is dedicated to Frank R. Giordano, without whose support and confidence I would not be involved in the honorable profession of teaching. It is dedicated to my dear friends, Jack and Denise Clemons, who gave openly of their hearts and their home throughout my studies. It is dedicated to my parents, , who sacrificed many things in the belief that a child's education should be such a high priority. Thank you.

Most importantly, this work is dedicated as a small measure of my absolute devotion to , my wife and lifelong partner, without whose patience, love, encouragement, and sacrifice, I surely could not have endured this experience. Finally, it is dedicated with love to both of my children, , in the hope that it might illustrate two bedrock principles of my personal philosophy: the final measure of your life is not in the amount of wealth and possessions accumulated, but rather in whether you have made a difference in this world with your existence; and, you can achieve anything in life, regardless of the obstacles those of lesser ambition will create, if you believe in yourself.

Acknowledgement

I wish to gratefully acknowledge the support of the Mathematical Sciences Laboratory at the U.S. Military Academy, Department of Mathematical Sciences, West Point, New York, for the generous use of their facilities in support of the computational testing performed herein. Additionally, I would like to acknowledge the support of the Optimization and Simulation Laboratory at Virginia Tech, Blacksburg, Virginia, for the use of their computing facilities during the early stages of developing this reformulation strategy. Finally, I wish to thank Mr. Ramesh Raman at the GAMS Development Corporation, for his much appreciated assistance in running the $n = 50$ symmetric test problem instance for computational testing.

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

It is well known in discrete optimization literature that in order to be able to solve resonably sized instances of challenging classes of problems, two particular features must come into play. First, a good model of the problem must be constructed in the sense that it affords a tight underlying linear programming representation, and second, any inherent special structures must be exploited, both in the process of model formulation and in algorithmic developments.

Prior to the 1970s when researchers began to employ elements of polyhedral study to problems in this class, two primary approaches for solving combinatorial optimization problems dominated the literature: *branch-and-bound*, an enumerative divide-and-conquer approach; and a *cutting plane* approach, a procedure that generates additional constraints and imposes these restrictions on the region of feasible solutions to the linear programming relaxation of a given problem. These two techniques were considered state-of-the-art for solving this class of difficult problems. Although both techniques held promise for the eventual defeat of successively larger and larger problem instances, each possessed inherent shortcomings that would not be overcome until several years later.

Before presenting an historical perspective on the developments in this field in order to further motivate our study, we first provide a statement of the research objective of this dissertation.

1.1. Statement of Research Scope and Contributions

Although many large–scale optimization problems have succumbed to improved algorithms and advances in computing technologies, the class of problems known as 0-1 mixed-integer programs

(MIP) in discrete optimization remain steadfast in their challenge. 0-1 MIPs arise in a plethora of real world problems including optimal cargo loading for container vessels, airline crew scheduling, telecommunications network design, plant location, capital rationing and strategic forest planning models. This research is primarily motivated by the prevalence of such problems in both Operations Research and Industrial Engineering applications.

The most promising of current techniques for solving 0-1 mixed-integer programs focus on algebraically manipulating the constraints of the model so that the geometric region defined by the constraints of the linear programming representation is as constricted, or as *tight*, as possible with respect to the smallest polyhedral geometric region that encapsulates all integer feasible solutions. If one were able to tighten the constraints to such a degree that the relaxed feasible region coincided with the integer feasible region, known as the *convex hull* of integer feasible solutions, the ideal would be achieved: solving the linear programming representation would solve the mixed-integer problem as well. Practicality and experience has dictated that a tightening process of this nature should be applied in degrees, depending upon the specifics of the problem being addressed.

This dissertation addresses a novel method for exploiting special structures of discrete optimization problems in order to construct tight relaxations that would enable large practical instances of difficult combinatorial problems to be solved more effectively. Problems having the type of structures analyzed herein arise in many applications such as job scheduling and sequencing in production/manufacturing floor-shops, engineering design contexts, personnel and crew allocation and scheduling problems, facilities planning and product distibution problems, and in the routing of service vehicles, among many other situations. The new hierachy of relaxations presented exhibits how to exploit the mathematical structure of such models in order to derive powerful solution techniques that would be capable of prescribing practical, economically attractive decisions. Tools of this type can be very useful in making both long-term strategic as well as day-to-day tactical operational decisions.

The new hierarchy presented in this study extends and subsumes the Reformulation–Linearization Technique (RLT) of Sherali and Adams (1989, 1990), currently state-of-the-art in this field, and is particularly designed to exploit explicit or implicit special structures defined by the constraints

of a problem in the following manner. By identifying specific classes of multiplicative factors that can be applied to the original mathematical formulation of the problem, the original problem is reformulated as an equivalent polynomial programming problem. Subsequently, this resulting problem is linearized to produce a tighter relaxation in a higher dimensional space created by a unique substitution of variables. The hierarchy is defined by selecting *a priori* the number and type of multiplicative factors to be applied to the constraint set. The *level of relaxation*, equal to the highest degree of the polynomial terms used in the multiplicative factors, is thus derived. This general framework permits us to generate an explicit hierarchical sequence of tighter relaxations leading up to the convex hull representation in which the final level equals the number of 0-1 integer variables involved in the problem. Although several other researchers have proposed reformulation-linearization techniques that are variations of the original Sherali-Adams (1989, 1990) technique, they remain, for the most part, existential in nature because of the difficulty in providing explicit algebraic representations that can be subsequently used in practice to derive other valid constraints that further assist in tightening the problem formulation.

The importance of this research becomes apparent when one realizes that *all* commercially available software packages currently utilize the linear programming relaxation of a 0-1 mixed-integer problem to guide the search of branch-and-bound algorithms through the binary branching tree. As demonstrated in the dissertation, for certain applications of this new hierarchy, not only are several levels of this branching process resolved at the very first node, but an implicit pruning of the branching tree also occurs. Thus, for these applications, not only is the initial linear programming representation tightened, yielding sharper bounds on the objective function value, but the number of branches of the binary tree requiring searching is reduced. Future research in this area will attempt to extend the tenets of the new hierarchy to larger classes of integer programming problems, and to develop a decomposition scheme taking advantage of parallel computing architecture.

1.2. Historical perspective

1.2.1. Branch-and-bound. The general approach of branch-and-bound can be most simply described as follows. The space of all feasible solutions to a given problem is repeatedly partitioned

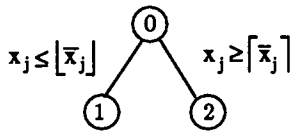

FIGURE 1. Dakin's variable branching dichotomy for nonintegral $\bar{x}_j$.

into smaller and smaller subsets, and a lower-bound (in the case of minimization) is calculated for the cost of the solutions in each subset. After each partitioning, those subsets having a lower bound that exceeds the cost of a known feasible solution are excluded from all further partitionings. The partitioning continues until a feasible solution is found such that its cost is no greater than the bound for any subset. For both pure integer programming (IP) and mixed integer programming (MIP) problems, the branch-and-bound approach first pioneered by Land and Doig (1960) dominated the scene. In this approach, the continuous relaxation of the IP or MIP is first solved. If the optimal solution satisfies the integrality conditions, the problem is solved. If not, then two new subproblems are created based on some fractional integer variable. For one subproblem, the chosen variable is restricted to its next lower integer value, and for the other subproblem, the variable is restricted to its next higher integer value. Each of these new subproblems is then solved, and the process is repeated. The original problem differs from each of the subproblems only in their respective domains of optimization. Branching in this manner forces fractional solutions to become infeasible, thus separating them from the feasible region, and inevitably converging on an integer optimal solution, assuming that one exists and that the integer restricted variables are bounded. Their methodology increased in popularity dramatically after Little, Murty, Sweeney and Karel (1963) applied it to the Traveling Salesman Problem (TSP), demonstrating the effectiveness of controlled enumeration on large, difficult problems.

Although Balas (1965) gave the first implicit enumeration algorithm for general 0-1 integer programs, Dakin (1965) proposed the now commonly used variant of Land and Doig's approach for both pure and mixed-integer 0-1 programs. His methodology for branching used the variable dichotomy illustrated in Figure 1. Unlike the tree structure created by Land and Doig's technique, for a chosen fractional integer variable, $\bar{x}_j$, rounding this variable to its next lower integer value $\lfloor \bar{x}_j \rfloor$ for one branch, and its next higher integer value $\lceil \bar{x}_j \rceil$ for another creates a *binary* tree.

Convergence to an optimal solution meeting the integrality conditions of a problem under the assumptions of existence and boundedness as noted previously, resolves the values of binary integer variables in the same fashion as Land and Doig's technique for general integer variables. Later, Driebeek (1966) introduced penalties on the branchings that were further sharpened by Tomlin (1971).

Although branch-and-bound performed reasonably well on limited sized problem formulations (See Lawler and Wood (1966), and Mitra (1973)), the number of subproblems that were required to be solved during the execution of the algorithm increased exponentially with the size of the problem, thus inhibiting the solution of even moderate sized practical integer problems. Additionally, the amount of temporary computer storage required during the execution of the branch-and-bound process, which is related to the maximum cardinality of the set of active nodes on the branching tree, i. e., the number of active subproblems, exceeded machine capacity as problem size increased. It was therefore understood that further advances would need to curtail this explosion associated with an increase in problem size.

Strategies were consequently developed to limit the growth of the binary search tree created by the general branch-and-bound algorithm, and hence, reduce the computational burden imposed by any sizable problem instance. Early on, these strategies focused exclusively on algorithmic aspects of branch-and-bound, such as deciding which variable to branch on, and choosing which subproblem to solve, among others (Beale (1979)). However, none of the commercial and academic codes developed during this period attempted to resolve the one inherent weakness indigenous to branch-and-bound: to set the lower (or upper) bound for each subproblem, branch-and-bound had to solve a naive linear programming relaxation. The *integrality gap* that existed between the value of this naive relaxation, $v(LP)$, and the optimal solution to the integral problem $v(IP)$, defined by $|v(LP) - v(IP)|$, was frequently far too large relative to $v(IP)$. The larger this (percentage) gap, the more branchings were necessary to resolve the values of integer variables. So, it seemed as if two options were at-hand: reduce the size of the gap prior to handing the problem over to a branch-and-bound algorithm, or reduce the size of the gap during the runtime of branch-and-bound, "on the fly", so to speak.

1.2.2. Cutting planes. Seeking to close the integrality gap during the branch-and-bound process, researchers looked to a strategy for generating cutting plane constraints as needed at particular nodes of the branching tree. An early class of cutting planes, introduced by Gomory (1960), served to validly separate the current subproblem linear programming relaxation solution $\bar{x}$ from the convex hull of feasible discrete solutions for all subsequent subproblems, thus achieving a certain tightening of the problem's feasible domain.

Although certain problems, such as set covering problems, were solved more effectively using cuts like Gomory's, cutting planes such as these possessed several shortcomings that detracted from their appeal. Gomory's cutting plane methods behaved very differently on quite similar problems (Gondran (1979)). Fulkerson, Nemhauser and Trotter (1973) illustrate two examples of relatively small-sized set covering problems that appeared to be quite intractable when attacked with these cutting planes. During the 1970's, there was practically no understanding as to why this effect should occur, other than the vague notion that the naive linear programming relaxation provided a poor approximation to the original integer program.

An additional source of difficulty with cutting planes was caused by optimizing over a subset of the original domain that was defined by adding a cutting plane constraint to a subproblem during the execution of a linear programming based branch-and-bound algorithm. Although such a cutting plane generated for a particular subproblem maintained its validity for all branching subproblems whose feasible regions are subsets of that restricted by the cutting plane, the validity of the constraint was not guaranteed for other subproblems residing on other distinct branchings. This localization of validity increases the amount of temporary storage required by a branch-and-bound algorithm that incorporates such cuts, and increases the overall computational burden by having to generate node-customized cuts. This is more of a burden on algorithms that use a breadth-first search scheme than those that incorporate either a depth-first search, or combination depth-first–breadth-first search scheme that is close to depth-first search in the manner of data storage.

Finally, Gomory's cutting planes also proved to be weak in the sense that they did little to reduce the size of the integrality gap for most problems. As an example of the failure of cutting planes, in an experiment conducted by Dakin (1966), the solution of a problem with 17 constraints

and 93 variables, 80 of which were constrained to be integers, took 7 minutes and 28 seconds on a KDF9 computer using branch-and-bound, while the Gomory mixed integer algorithm failed to find the solution in over 2000 iterations. It was becoming apparent that any cuts generated either prior to handing the problem to a branch-and-bound procedure or during its execution had to be deeper, imposing a more significant structural tightening of the linear programming feasible region. The integrality gap should be significantly reduced by any constraints that are added to the problem. The most ideal situation, of course, would have each additional constraint restrict the feasible region to such an extent that solving the associated linear programming relaxation for a given integer problem would yield the optimal solution to the integer problem

1.2.3. Balas' enumerative algorithm. During this same period, Balas (1965) proposed an enumerative approach tailored to the 0-1 integer programming problem that avoided linear programming relaxations entirely. For each subproblem that was created by branching, Balas partitioned the problem variables into three classes. One class contained those variables set to 0; the second contained those set to 1; and the third contained those that remained free to assume either the value of 0 or 1. Assuming that all cost coefficients were positive, a solution obtained by setting all variables in the last class to 0 was clearly optimal (zero completion). If this solution was not feasible, additive logic was applied to the objective function and the constraints in order to restrict variables to 0 or 1 by virtue of feasibility considerations, or based on obtaining further improvements in the objective function value. Branching was effected by forming one subproblem in which a specific free variable is set to 1, and a second subproblem with the same variable set to 0.

An essential algebraic feature of Balas' algorithm was the use of the foregoing logical tests to fix variables, and hence tighten the continuous relaxation, as well as to compute lower bounds by examining sets of free variables that must contain members that take on a value of 1 for problem feasibility. The difference between Land and Doig's approach and that of Balas is that the bounds determined by Balas' algorithm were not based on linear programming relaxations; yet, the linear programming relaxations were implicitly strengthened via the 0-1 variable fixing process. Using Balas' algorithm, Freeman (1965) reported favorable results on ten small-sized fixed-charge problems ranging from 4 inequalities in 14 binary variables to 6 inequalities in 20 binary variables.

Later, Geoffrion (1969) exhibited that Balas' additive algorithm could be improved by coordinating this with the solution of linear programming relaxations at judiciously chosen nodes. This laid the foundation of linear programming based branch-and-bound with a preprocessor to tighten the linear programming relaxation prior to its solution.

1.3. Polyhedral developments

In the 1970's, the focus of interest for combinatorial optimization — originally tending exclusively to algorithmic aspects of these problems — shifted to an investigation of the facial structure of *polyhedra* associated with these problems. Research into *polyhedral theory* was motivated by the desire to obtain in a systematic manner tighter problem formulations rather than simply mathematically correct formulations of such integer programming problems. New theorems and algorithms emerged that identified classes of constraints that could be generated as needed within a branch-and-bound environment. However, these cuts possessed an important property distinguishing them from earlier cutting planes such as Gomory's: they frequently defined *facets* of the problem's underlying polyhedral structure or substructure. Such inequalities are called *facets*, or *facet defining*, because, by definition, they define a hyperplane which itersects an integer polyhedron $P_I(A, b)$, defined by the constraints of the linear system $Ax \geq b$, in a face of dimension one less than the dimension of $P_I(A, b)$. Facets belong to a system of linear inequalities that minimally define the convex hull of feasible solutions to an integer programming problem.

This new shift in approach, pioneered by Padberg (1973), recognized the much stronger structural properties present in combinatorial optimization problems such as the *set packing* problem:

$$\begin{aligned} \text{(SP)} \qquad & \max \quad cx \\ & \text{subject to:} \quad Ax \leq e, \\ & \qquad x_j \in B = \{0,1\} \qquad \forall\, j \in N = \{1,\ldots,n\}, \end{aligned}$$

where A is an $m \times n$ matrix of zeros and ones, c is an arbitrary n-vector and $e^T = (1,\ldots,1)$ is an m-vector; the equality constrained *set partitioning* variant of this problem obtained by replacing $Ax \leq e$ by $Ax = e$; and the *set covering* problem that restricts $Ax \geq e$. Thus, x^* represents a *packing* (respectively, partitioning, cover) if and only if $x^* \in B^n$ satisfies $Ax^* \leq e$ (respectively,

$Ax^* = e$, $Ax^* \geq e$).

Associated with the problem (SP) is a graph $G = (N, E)$ where a node $j \in N$ is defined for each column of A, and an edge $(i, j) \in E$ joins node i and node j if and only if column i and column j of A have at least one $+1$ entry in common. G is referred to as the *intersection graph.* A *clique* in a graph G is a maximal complete subgraph of G. Padberg (1973) demonstrated that the inequality

$$\sum_{j \in K} x_j \leq 1, \tag{1}$$

where $K \subseteq N$, corresponds to a facet of the convex hull of integer feasible solutions to (SP) if and only if K is the node set of a clique in G. A second set of facets were also identified as originating from odd length cycles without chords of the same intersection graph, where a cycle without chords in a graph G satisfies the additional requirement that no edges of G connect two non-consecutive nodes of the cycle in question.

Padberg (1973) also demonstrated that cutting planes did not possess a facetial property in general, and were not even guaranteed to intersect the convex hull of integer solutions. Although a complete polyhedral description for most interger programming problems proved elusive, branch-and-bound procedures incorporating even *partial* convex hull descriptions provided by facetial constraints tended to perform better, generating fewer active nodes in the enumeration process and terminating faster because of improved linear programming relaxation bounds. Padberg and Hong (1977) applied this methodology with excellent results to a 318-city traveling salesman problem (TSP), originally reported by Lin and Kernighan (1973), with $m = 318$ equality constraints and $n = 50,403$ relaxed 0-1 variables. Adding 183 automatically generated facetial inequalities and solving the augmented, tighter linear program to optimality took 37 minutes of CPU time on an IBM 370/168 MVS, and reduced a previously reported integrality gap of 2.583 inches to 0.112 inches, moving the linear programming relaxation solution to within 0.25% of optimum.

The success of this polyhedral approach created a spark of excitement as researchers recognized the correlation between combinatorial optimization problem formulations and natural graph theoretical problems. The research that followed Padberg (1973) extended both an understanding of polyhedra for specially structured problems and the class of problems for which specially struc-

tured facetial cuts could be readily identified and generated. Grötschel and Padberg (1975) identified several new inequalities for the linear programming formulation of the asymmetric traveling salesman problem (ATSP) that yielded a sharper characterization of the underlying polytope.

The investigation of polyhedral characteristics of various combinatorial optimization problems began to reveal that there existed characteristics shared in common between seemingly unrelated problems. Padberg and Rao (1974) showed that the traveling salesman polytope, as well as the assignment problem, and the edge-matching problem on complete graphs all had a diameter, as defined below, less than or equal to two. The diameter of a polytope associated with a combinatorial optimization problem was generally accepted as a measure of the computational complexity of the problem, and is defined in the following manner. Let $X = \{x \in R^n : Ax = e,\ x \geq 0\}$, where A is any $m \times n$ matrix of zeros and ones and $e^T = (1, \ldots, 1)$ is an m-vector. The system $Ax = e$ defines a manifold in R^n. Denoting by X_I the convex hull of integer vertices of X, the diameter $\delta(X_I)$ of the polytope X_I associated with this convex hull is defined in the following way: A path on X_I between two vertices x^1 and x^2 is a sequence of distinct vertices $(x^{1,0}, \ldots, x^{1,p})$, with $x^{1,0} = x^1$ and $x^{1,p} = x^2$, such that every pair of vertices $x^{1,i}, x^{1,i+1}$, $i = 0, 1, \ldots, p-1$, is connected by an edge of the polytope X_I. The length of such a path is defined to be equal to p. The edge-distance $d(x^1, x^2)$ between x^1 and x^2 is then defined to be the length of a shortest path between x^1 and x^2. The diameter $\delta(X_I)$ of the polytope X_I is the longest edge-distance between any pair of vertices of X_I, *i.e.*, $\delta(X_I) = \max_{x^1, x^2 \in \text{vert} X_I} d(x^1, x^2)$. Padberg and Rao (1974) demonstrated that $\delta(X_I) \leq 2$.

Besides the interesting fact that the diameters of these polytopes, defined in the precise manner above, were independent of the dimensionality of their respective associated problem formulations, these results were significant in that the traveling salesman polytope was demonstrated to share a characteristic in common with ones for which there already existed "good" algorithms. This provided hope that some generalization of solution approaches might be possible for a much larger class of problems, including the general zero-one programming problem.

Apart from specifically identifying classes of facets, Bradley, Hammer and Wolsey (1974) maintained an algebraic orientation with regard to seeking tighter formulations. For a given inequality

with 0-1 variables, say

$$\sum_{j=1}^{n} a_j x_j \leq a_0, \qquad x_j = 0, 1, \tag{2}$$

they demonstrated that there can exist other "equivalent" inequalities given by $\sum_{j=1}^{n} b_j x_j \leq b_0$ with x_j binary that have exactly the same 0-1 feasible solutions as the original one, but are "stronger" than the original inequality. In their development, they considered only inequalities that satisfied the three *normalization* conditions (a) $b_j \geq 0$ for all $j = 1, \ldots, n$, (b) all infeasible points X satisfy $\sum_{j=1}^{n} b_j x_j \geq b_0 + 1$, and (c) $b_1 \geq b_2 \geq \cdots \geq b_n$. This set of equivalent inequalities defines a polytope over which the construction of an equivalent inequality having minimum coefficients can be accomplished via a linear program. Two definitions are necessary to illustrate how the linear program is formulated.

An n-vector x^* with binary coefficients is a *ceiling point* of (2) if

(1) $\sum_{j=1}^{n} a_j x_j^* \leq a_0$;

(2) if $x_t^* = 0$, then $\sum_{j=1}^{n} a_j x_j^* + a_t \geq a_0 + 1$; and,

(3) if $x_t^* = 0$ and $x_{t+1}^* = 1$, then $\sum_{j=1}^{n} a_j x_j^* + a_t - a_{t+1} \geq a_0 + 1$.

The set $\{j : x_j^* = 1\}$ is called a *ceiling* of (2). Thus, if every ceiling point of (2) is a feasible point of the equivalent inequality, then the equivalent inequality is a relaxation of (2).

An n-vector x^* with binary coefficients is a *roof point* of (2) if

(1) $\sum_{j=1}^{n} a_j x_j^* \geq a_0 + 1$;

(2) if $x_t^* = 1$, then $\sum_{j=1}^{n} a_j x_j^* - a_t \leq a_0$; and,

(3) if $x_t^* = 1$ and $x_{t+1}^* = 0$, then $\sum_{j=1}^{n} a_j x_j^* - a_t + a_{t+1} \leq a_0$.

The set $\{j : x_j^* = 1\}$ is called a *roof* of (2). Thus, if every roof point of (2) is an infeasible point of the equivalent inequality, then the equivalent inequality is a restriction of (2).

Let C and R represent the collection of ceilings and roofs of (2). An inequality is equivalent to (2) if and only if (a) $\sum_{j \in \tilde{C}} b_j \leq b_0$ for all $\tilde{C} \in C$, (b) $\sum_{j \in \tilde{R}} b_j \geq b_0 + 1$ for all $\tilde{R} \in R$, and (c) $b_1 \geq \cdots \geq b_n \geq 0$. Using these definitions, a "minimum" equivalent inequality can be constructed

by first determining all the roofs and ceilings and then solving the linear program

$$\begin{aligned}
\text{minimize } & \sum_{j=0}^{n} b_j, \\
\text{subject to: } & \sum_{j \in \tilde{C}} b_j \leq b_0 && \text{for all } \tilde{C} \in C, \\
& \sum_{j \in \tilde{R}} b_j \geq b_0 && \text{for all } \tilde{R} \in R, \\
& b_1 \geq b_2 \geq \cdots \geq b_n \geq 0; && b_0 \geq 0.
\end{aligned}$$

A minimum equivalent inequality with integer coefficients is an optimal solution to the associated integer programming problem. A "minimum inequality" defined in this manner separates feasible and infeasible points in a "strongest" possible way, and hence was of interest in developing cutting plane algorithms, independent of the insights provided by polyhedral theory.

Notice that the linear objective function in the linear program above actually represents an L_1 norm. If all the feasible 0-1 points satisfy $\sum_{i=1}^{n} b_i x_i \leq b_0$ and all the infeasible 0-1 points satisfy $\sum_{i=1}^{n} b_i x_i \geq b_0 + 1$, one way to define a strongest equivalent inequality is to maximize the distance between these two inequalities. By defining the distance between these two inequalities as $1/\sum_{j=0}^{n} |b_i|$, the approach taken by Bradley, et al. (1974) encompasses the idea of strengthening a cutting plane by maximizing a distance measure between the original inequality and a stronger inequality introduced earlier by Bowman and Nemhauser (1971). Sherali and Alameddine (1992) implemented a similar Euclidean distance technique in a constraint filtering strategy to reduce the size of the bounding linear programs generated by **RLT0** for bilinear programming problems. In their approach, if the distance between the current solution and a particular hyperplane associated with an inequality constraint exceeds a specified tolerance, the constraint can be deleted for the remainder of decendent subproblems. Recently, Boyd (1994) has adapted this methodology to generate a class of deep cutting planes known as *Fenchel cuts* to enhance the *separation* problem associated with using Lagrangian relaxations to guide integer programming problems: find an inequality that is valid for the underlying integer programming problem but violated by the optimal solution of the linear programming relaxation. Computational results appear to indicate that with respect to the number of cuts generated and the extent to which the linear programming–

integer programming gap is closed, Fenchel cuts appear as effective as facets of the individual knapsack polyhedra.

Brearley, Mitra and Williams (1975) extended the early foray into *preprocessing* a given problem formulation, although they did not incorporate constraint generation within their approach. Their main emphasis was to reduce the dimension of the original problem by deleting rows and columns using logical tests, so as to reduce the time required to solve the associated linear programming relaxation and simultaneously limit computer storage requirements. They also prescribed a methodology to identify and exploit generalized upper bounding (GUB) row structures by identifying sets of variables in conjunction with their corresponding constraint relationships so that the total number of GUB-type constraints is maximized.

1.3.1. Lifting lower dimensional facets. Paralleling efforts to identify polyhedral characteristics of integer programming problems during the 1970's were attempts to extract and exploit *implicit* structures from these problems. These were extensions of an observation first noted by Pollatschek (1970) with regard to the structure afforded by independence systems: it is possible to obtain facets (valid inequalities) of the convex hull of binary integer feasible solutions to a general 0-1 program, by *lifting* facets and valid inequalities of lower dimensional polytopes. Specifically, if X_I is a feasible solution set of any 0-1 program in $N = \{1, \ldots, n\}$ variables, for every subset N' of N, $\emptyset \subseteq N' \subseteq N$, define:

$$G_I(N') = \{x \in X_I : x_i = 0,\ i \in N \backslash N'\}.$$

Then, given a valid facet (valid inequality) of the convex hull of this lower dimensional polyhedron, $\text{conv}(G_I(N'))$, defined by

$$\sum_{j \in N'} b_j x_j \leq b_0, \tag{3}$$

this lower dimensional facet can be sequentially *lifted* into a facet (valid inequality) of $\text{conv}(X_I)$ defined by

$$\sum_{j \in N'} b_j x_j + \sum_{j \in N \backslash N'} b_j x_j \leq b_0. \tag{4}$$

This lifting is accomplished by introducing variables x_j, $j \in N \backslash N'$, in a predefined order, and computing each corresponding lifting coefficient $b_j \geq 0$ in such a way as to insure (4) is as tight as possible while insuring that the constraint being lifted remains valid at each step in the process. If the coefficients of x_j, $j \in N \backslash N'$, are determined simultaneously, then the process is referred to as one of *simultaneous lifting*.

Padberg (1973) introduced a procedure for calculating the sequentially lifted coefficients for these constraints that proved easy to implement within branch-and-bound methods. His procedure was first applied to the set packing polytope, and then extended to any 0-1 linear program whose coefficient matrix was nonnegative. The same procedure was used for other 0-1 problems containing sufficient explicit structure to facilitate this approach. Among these were the vertex packing problem and independence systems (Nemhauser and Trotter (1974), and Trotter (1975)), the knapsack polytope (Balas (1975)), regular 0-1 polytopes (Hammer, Johnson and Peled (1975)), strong covers of 0-1 inequalities (Wolsey (1975)), minimal covers of the knapsack polytope (Balas and Zemel (1978)), and generalized minimal covers of GUB constrained knapsack problems (Sherali and Lee (1992)). Zemel (1978) generalized the sequential lifting procedure to the class of 0-1 programming problems. It is worth noting that the procedure of Zemel (1978) and Sherali and Lee (1992) actually produce facetial inequalities of the respective underlying convex hull polytopes in polynomial time. As pointed out by Padberg (1979), while these problems appeared to be significantly different from each other, packing, covering, and knapsack problems can ultimately be viewed as optimization problems over undirected graphs. This observation facilitated insights into the underlying structure of these problems that might have gone unnoticed.

All of the applications of lifting mentioned so far used a common technique of introducing new variables $b_j x_j$, $j \in N \backslash N'$, one at a time to derive some lifted valid inequality. As a direct result, the lifted variable coefficients obtained in this sequential manner turned out to be integer valued. The facets obtained by *sequential lifting* depended upon the sequence in which the variables were introduced. As noted by Zemel (1978), even when all possible sequences of introducing new variables were followed, the family of facets of the form (4) was still not exhausted. A generalized procedure in which sets of variables from the subset $N \backslash N'$ were simultaneously introduced was proposed by Zemel (1978). This *simultaneous lifting* was found to generate additional facets un-

obtainable by sequential lifting, i.e., those facets with *fractional* coefficients. However, because the number of integer programs one has to solve to generate all the facets of a given 0-1 polytope is prohibitively large, it was questionable whether this generalized approach would prove computationally advantageous. Zemel (1978) observed that the procedure becomes more effective when it is applied to problems whose special structure helps to cut down the number of integer programs required to be solved, or simplifies the solution. The necessity to exploit special structures, when present, again became apparent.

1.4. Algorithmic advances

By the end of 1979, integer programming codes took advantage of many of the innovations inspired by polyhedral theory during this decade, so that serial computations were about 10 times faster on the same hardware as they were in 1974 (Beale (1979)). Consequently, the scope of integer programming applications increased; but not dramatically so, because the size of an integer programming problem measured in numbers of constraints, variables, and nonzero coefficients remained a poor guide as to its difficulty. Beale (1979) concluded that unless the number of integer variables is very small ($n \leq 20$), it was important that the continuous optimum should provide a fair guide to the integer solution, or at least the value of the objective function in the relaxation should approximate the best integer solution. However, even the incorporation of deep cuts within a branch-and-bound methodology failed to remove some misgivings. Sometimes, cuts would destroy the matrix structure present in the problem prior to their addition, hence inducing difficulties. As Milotis (1976) showed for the symmetric traveling salesman problem, cuts such as Gomory's tended to be dense, with very few zero coefficients. Efforts to preserve sparsity while incorporating these cuts would therefore be self-defeating. Additionally, weaker cuts such as those of Gomory, were discarded when they exhibited slack, while facetial cut constraints tended to be retained in storage throughout calculations. Depending upon the number of cuts generated for specific subproblems of a branching tree, this additional storage requirement could present a significant obstacle in tackling large–scale problems on machines employing either a single memory system, or a parallel environment possessing global shared memory architecture. This point underscores the importance of finding an approach that generates facetial cuts valid

for the *complete* polytope, and not simply a subset of the original feasible region, both from a standpoint of minimizing storage requirements during problem execution, and providing as tight as possible initial problem formulation prior to execution.

Figure 2 illustrates the structure of one algorithmic approach proposed by Milotis (1978) for the traveling salesman problem that is typical of the class of algorithms that incorporated cutting planes. Similar to earlier work (Milotis (1976)), the starting set of constraints is a subset of the set of constraints that completely describes the problem, i.e., the assignment constraints $\sum_i x_{ij} = 1 \quad \forall \quad j$, and $\sum_j x_{ij} = 1 \quad \forall \quad i$, with $0 \leq x_{ij} \leq 1$. The linear programming relaxation is solved repeatedly, but instead of branching upon some fractionating variable, cutting planes are used to drive a fractional solution to an integer one. When an integer solution is found, it is tested for feasibility (*i.e.*, a feasible Hamiltonian circuit). If the solution is feasible, it is optimal as well and the algorithm terminates. Otherwise, violated constraints are generated, added to the current set of constraints, and the reoptimization process is reiterated. The omitted constraints are generated automatically as needed, and the reoptimization takes place on the same computer run. Additionally, and somewhat unusually, the computations for this particular algorithm were performed in integer arithmetic. This was achieved by storing the determinant (denoted by IR in Figure 2) of each linear programming basis separately and multiplying the actual inverse and all the vectors associated with it, by the value of the determinant. Martin (1966) had previously used a slightly different starting formulation with no upper bounds of 1 on the relaxed binary variables, permitting two city subtours to also occur. Notice in the flowchart of Figure 2 the lack of problem preprocessing advocated by Brearley, Mitra and Williams (1975) that would later prove of fundamental importance in tightening a given problem's formulation.

1.5. Hybrid algorithms

Enthusiasm over identifying and exploiting underlying polyhedral structures of combinatorial optimization problems showed no signs of waning in the 1980's. This period ushered in several seminal papers that presented comprehensive computational strategies incorporating ideas of exploiting special structures present in problem formulations. Most, if not all, of the large-scale mathematical programming systems designed to solve 0-1 mixed-integer programs, such as

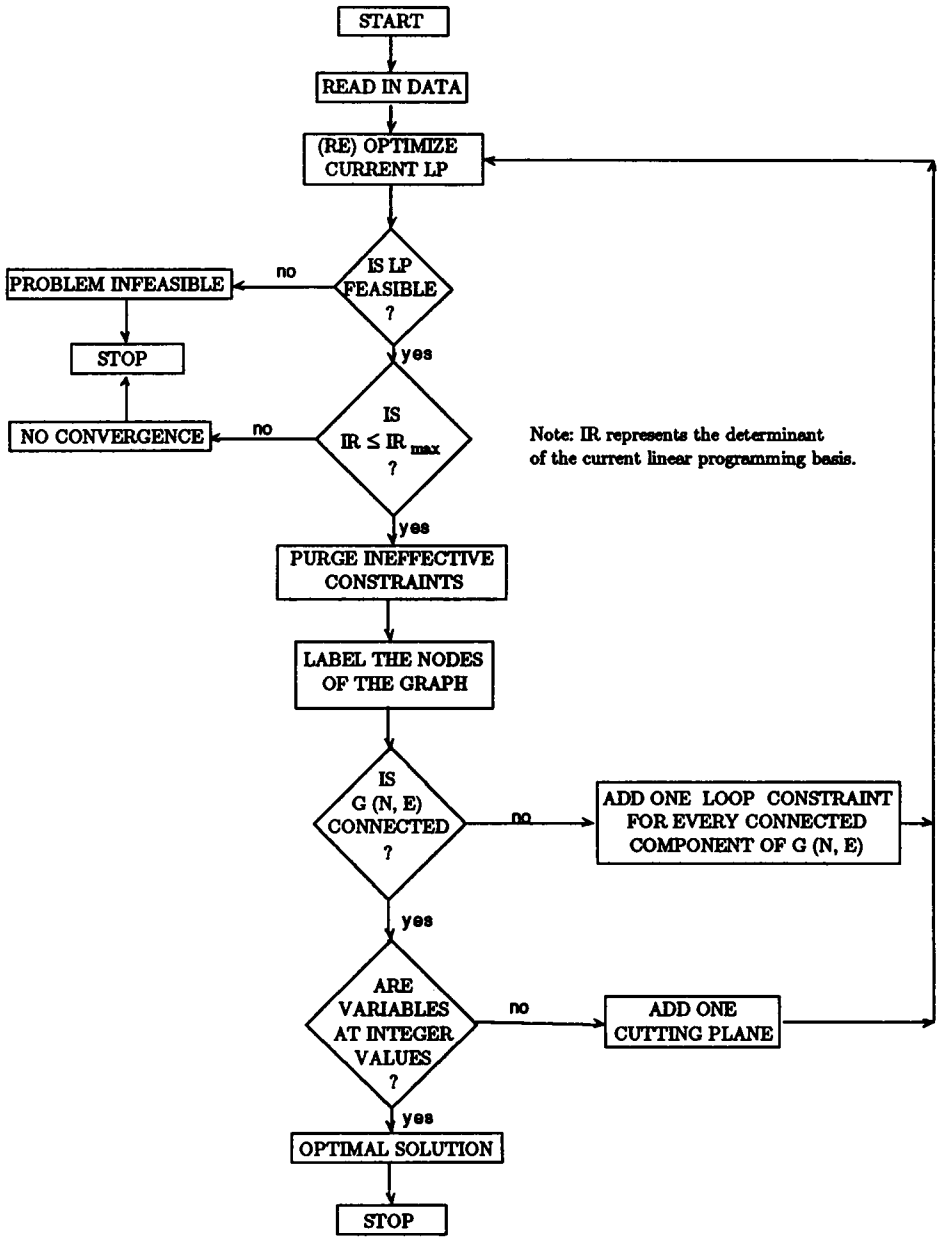

FIGURE 2. Milotis' algorithm with cutting planes.

MPSX-MIP/370, APEX-III, UMPIRE, FMPS, LP 6000, etc., used branch-and-bound algorithms based on exploring a sequence of linear programming relaxations in which only a subset of the 0-1 variables were fixed and all other variables treated as continuous. However, for large-scale problems with many 0-1 variables and a highly structured coefficient matrix on these variables, running times in finding and proving optimality could be excessive.

To take advantage of the binary nature of these type problems, Johnson and Suhl (1980) introduced a hybrid algorithm to solve large scale zero-one integer programming problems that combined branch-and-bound, *implicit enumeration*, and an updated cutting plane generation scheme. One salient feature of their approach was their use of implicit enumeration. As opposed to solving a sequence of linear programming relaxation problems, enumerative algorithms solve a sequence of problems in which various 0-1 completions of partial solutions are explored. Let S denote the index set of those 0-1 variables assigned a value at any particular node of the branching tree, and $F(S)$ the index set of free variables, *i.e.*, those 0-1 variables not assigned a value. Then, to S corresponds the problem

$$\text{(JS(P(S)))} \qquad \text{minimize} \quad cx + \sum_{j \in F(S)} d_j y_j + z(S)$$

$$\text{subject to:} \quad Ax + \sum_{j \in F(S)} B_j y_j (\leq \text{ or } \geq) b(S),$$

$$x \geq 0,\ y_j \in \{0,1\} \qquad \text{for} \quad j \in F(S).$$

Within the branch-and-bound environment, implicit enumeration uses logical tests on y_j, $j \in F(S)$, as its main branch fathoming device. These logical tests are used to show that at a given node of the branching tree, assigning the opposite value to a 0-1 variable in $F(S)$ results in a fathomed problem. This type of approach is particularly attractive for problem formulations in which the number of 0-1 variables greatly exceeds the number of continuous variables.

For general 0-1 mixed integer programs, Johnson and Suhl (1980) applied the principles of implicit enumeration using a new style of cut constraints called Benders' cuts. More specifically,

consider an MIP of the form

$$\text{(JS(P))} \qquad \begin{aligned} \text{minimize} \quad & z = cx + dy \\ \text{subject to:} \quad & Ax + By(\leq \text{ or } \geq)b, \\ & x \geq 0,\ y \quad \text{binary}, \end{aligned}$$

where all vectors and matrices have real coefficients and have the proper dimensions. Typical Benders' cutting planes are generated in the following fashion. Let y^* be any zero-one vector with the correct dimension and z^* the value of any incumbent solution of JS(P). Upon substitution of y^* into JS(P), the LP problem

$$\text{(JS(LP(y*)))} \qquad \begin{aligned} \text{minimize} \quad & cx + dy^* \\ \text{subject to:} \quad & Ax(\leq \text{ or } \geq)b - By^*, \\ & x \geq 0, \end{aligned}$$

is obtained. Let $JS(PL)$ denote the polytope defined by the dual linear program of $JS(LP(y^*))$ which is independent of y^*. If $JS(LP(y^*))$ has an optimal solution, it is possible to derive a Benders' cutting plane $z^* \geq dy + \bar{u}(b - By)$ in 0-1 variables alone, where $\bar{u}$ is an extreme point of $JS(PL)$. If JS(LP(y*)) is infeasible, a cut of the form $\bar{v}(b - By) \leq 0$ is generated, where $\bar{v}$ is an extreme ray of $JS(PL)$. The first cut can be derived directly from the reduced costs of the optimal primal tableau of $JS(LP(y^*))$. Enumeration is then performed on the 0-1 variables in the following fashion. Again, let S be a partial solution of (5), and $y(S)$ the vector of zero completion of S. If $y(S)$ satisfies the pure constraints on y and the current Benders' cuts, $JS(LP(y(S)))$ is solved and a new cut is generated. This variation has an advantage over the original algorithm of Benders (1962) in that only one 0-1 problem has to be solved.

The disadvantages of using explicit Benders' cuts mirror those noted earlier for Gomory's cuts: (a) the cuts are usually dense, and adding them alters any special structure previously present in the formulation; and, (b) the number of cuts generated for a general MIP is large, imposing a memory storage burden on the algorithm. Primarily for these reasons, Johnson and Suhl (1980) used implicit Benders' cuts via reduced costs of linear programming relaxations at specific nodes of the branching tree. More precisely, instead of solving the LP-relaxation corresponding to a

partial solution S, $JS(LP(y(S)))$, wherein any free variables are set to zero (or using any other completion), they treated the free variables as *continuous variables* in the LP-relaxation, and retained logical tests together with side constraints. This is an important distinction. By solving the problem

$$\text{(JS(LP(S)))} \qquad \text{Minimize } cx + \sum_{j \in F(S)} d_j y_j$$

$$\text{subject to: } Ax + \sum_{j \in F(S)} B_j y_j (\leq \text{ or } \geq) b(S),$$

$$x \geq 0,\ 0 \leq y_j \leq 1,\ j \in F(S),$$

two advantages were gained. First, since the dual polytope $JS(PL)$ is independent of y^*, typical Benders' cuts can still be derived from $JS(LP(S))$, if desired. Second, $JS(LP(S))$ is a relaxation of its integer counterpart $JS(P(S))$, which is not true of $JS(LP(y(S)))$. If $JS(P(S))$ is infeasible at any node of the branch-and-bound, $JS(P(S))$ is fathomed. Similarly, if $JS(P(S))$ is integer, it is fathomed immediately. In case $JS(LP(y(S)))$ is infeasible, the only option is to generate an extreme ray Benders' cut and continue exploring $JS(P(S))$. Their numerical results indicated that the more frequently the linear programming relaxation is solved, the fewer the number of nodes that need to be generated to prove optimality. This further reinforces the importance of having a tight linear programming relaxation in-hand for use in estimating the appropriate lower/upper bounds. For many practical problems, it is not necessary or at least not economical to prove optimality. If some optimality tolerance $0 < \epsilon < 1$ is specified, then every time a better integer solution z^* is found, a new problem bound $z' = (1-\epsilon)z^*$ can be used to guide subsequent searches, with the algorithm terminating when the best found integer optimal solution is within $\epsilon \cdot 100$ percent from an optimal integer solution.

1.5.1. Cutting planes with TSP. Crowder and Padberg (1980) utilized another hybrid algorithm that combined linear programming based branch-and-bound and specialized cutting planes derived from the special structure provided by the zero-one integer programming formulation of the symmetric traveling salesman problem (TSP). Using this approach, they solved to optimality ten problems ranging from 48 to 318 cities. The cutting planes generated by exploiting the TSP special structure were facets of the underlying TSP polytope. Although several formula-

tions of the asymmetric traveling salesman program (ATSP) are presented in detail later in this dissertation, it is worthwhile to mention the specific zero-one programming formulation used by Crowder and Padberg (1980) to highlight their exploitation of three special structures afforded by TSP.

Let $G = (V, E)$ denote a complete, undirected graph having n nodes (representing cities to be visited by the salesman) and let the vector c, with components c_e for all edges $e \in E$, denote the symmetric distance table of the graph. In this sense, if node i and j are connected by edge e, then c_e represents the distance between these nodes. For any vector x having components x_e for $e \in E$, denote for any subset $W_i \subseteq V$, $\bigcup_i W_i = W$,

$$x(W_i) = \sum_{e \in E(W_i)} x_e,$$

where $(W_i, E(W_i))$ is a complete subgraph of G. Let A be the $n \times m$ node-edge incidence matrix of G. The symmetric TSP can then be formulated as

(TSP) minimize cx

(5) subject to: $Ax = 2$,

(6) $$\sum_{i=0}^{k} x(W_i) \leq |W_0| + \sum_{i=1}^{k} (|W_i| - 1) - \left\lceil \frac{1}{2} k \right\rceil,$$

(7) $$x \geq 0, \qquad x \text{ integer},$$

where 2 is a vector having n components, each equal to 2, $\lceil \cdot \rceil$ represents the rounding-up operation, and the sets W_i are proper subsets of V satisfying the conditions:

(8) $$|W_0 \cap W_i| = 1, \qquad i = 1, \ldots, k,$$

(9) $$|W_i - W_0| \geq 1, \qquad i = 1, \ldots, k,$$

(10) $$|W_i \cap W_j| = 0, \qquad 1 \leq i < j \leq k,$$

(11) $$k \text{ odd}.$$

This formulation is valid for all odd k. The constraints $Ax = 2$ insure that every node of G is met by exactly 2 edges in every tour. The edge-set $C = \bigcup_{i=0}^{k} E(W_i)$ is called a *comb* in G in which W_0 is known as a *handle*, and the subsets W_i, $i = 1, \ldots, k$, k odd, are called *teeth*. The inequality

(6) satisfying conditions (8)—(11) is called a *comb constraint.* An inequality (6) expressed as $x(W_i) \leq |W_i| - 1$ for all $W_i \subseteq V$, $2 \leq |W_i| \leq n-1$, is called a *subtour elimination constraint*, expressing the logical condition that tours in the graph of length less than n are not allowed. An inequality (6) is called a *2-matching constraint* if the conditions in both (8) and (9) hold as equalities. A comb is called a *Chvatal-comb* if (10) is dropped and (8) is required to hold as an equality.

Crowder and Padberg's algorithm proceeds as follows. The linear programming relaxation given by

$$\text{(12)} \qquad \text{Minimize } \{cx : \ Ax = 2, \qquad 0 \leq x \leq 1\}$$

is initialized by a starting tour obtained heuristically using the edge-exchange method of Lin and Kernighan (1973). Then, the algorithm proceeds by adjoining cuts in a *primal* fashion: If the next feasible solution to the current linear programming relaxation is the incidence vector of a tour, a usual pivot is executed. Alternatively, assuming that the next feasible solution is not identical to the current one, it is chopped off by some cutting plane that is satisfied by the current solution as an equality. To generate the required cutting plane, subroutines were called that attempted to identify subtour elimination, 2-matching, and comb constraints. If suitable constraints $Dy \leq d$ were identified, they were adjoined to the current linear programming relaxation, and a degenerate pivot was executed on this enlarged problem. Since *all* tours satisfy these new constraints but a portion of the feasible region of (12) has been chopped off, the linear programming relaxation has been "tightened." Finiteness was guaranteed because the number of subtour elimination and comb constraints are finite. If the situation was encountered where a suitable constraint could not be found, either the incidence vector of the current linear program represented an optimal tour to TSP, or the current linear programming relaxation was solved to optimality to obtain a true lower bound on the minimum tour length.

As a final step, when the cutting plane routine exited without an optimal tour, it attempted to reduce the problem size by fixing variables at either zero or one based on their reduced costs, $\overline{C}_j$, obtained by the optimal linear programming relaxation. That is, since the reduced cost of all nonbasic variables with value zero at the linear programming optimum satisfies $\overline{C}_j \geq 0$,

a nonbasic variable x_j could be fixed at value zero if $\overline{C}_j \geq \Delta$, where Δ is defined as the gap between the best tour obtained and the optimal value of the objective function of the current linear programming relaxation. Similarly, since the reduced cost of any nonbasic variable with value one at the linear programming optimum satisfy $\overline{C}_j \leq 0$, a nonbasic variable x_j can be fixed at value one if $|-\overline{C}_j| \geq \Delta$ holds.

At an iteration where optimality was not detected, the output of this cutting plane procedure was a linear programming problem of the form:

$$\text{Minimize } \{c^r y : A^r y = b,\ Dy \leq d,\ 0 \leq y \leq 1\}, \tag{13}$$

where c^r is a vector with m_r components unable to be fixed at zero or one, A^r is the corresponding $n \times m_r$ submatrix of A, b is an n-vector with components equal to 0, 1, or 2, D is the matrix whose rows corresponding to the cutting plane constraints generated, and d is a vector that has been adjusted from its original values to its corresponding new right-hand side values that result due to variable fixing. At this stage, another procedure found an optimal solution to (13), which was then used as input to a branch-and-bound procedure to find a feasible 0-1 solution to the problem (13). If this 0-1 solution possessed subtours, further subtour elimination constraints were generated and appended to (13), and passed on to the main linear program (12). Problem (12) was then re-optimized starting with the feasible basis of the previously solved linear program, and the branch-and-bound procedure would then be continued. Iterating in this fashion, the overall program halted after finding a minimum-length tour to TSP.

In general, as shown in Table 1, their cutting plane approach produced very few comb constraints, with a vast majority of the cutting planes being comprised of subtour elimination constraints and 2-matching constraints. To the surprise of Crowder and Padberg, the entire procedure never iterated more than three times in order to find an optimal tour, and the largest problem tested, involving 318 cities, required only 216 nodes within the two executed branch-and-bound procedures. Their results emphasized the suitability of facet-defining cutting planes for the purpose of proving optimality in hard and difficult combinatorial optimization problems. Perhaps more importantly, not only was the total number of nodes generated by the branch-and-bound procedure small in the presence of these specially structured cutting planes, but the *incremental*

Problem	No. cities	Subtour	2-Matching	Comb	Total
GRO48	48	21	16	1	38
TOM57	57	16	16	0	32
KRO124	100	54	32	1	87
KRO125	100	40	28	2	70
KRO126	100	43	37	3	83
KRO127	100	55	36	1	92
KRO128	100	45	57	1	103
GRO120	120	51	28	0	79
KNU121	121	18	0	0	18
LIN318	318	157	20	0	177
Total		500	270	9	779
Percent		64.3	34.6	1.1	100

TABLE 1. Cutting planes for TSP.

number of nodes inspected by the branch-and-bound procedure in order to prove optimality once an optimal solution to the linear programming relaxation was found was also uncharacteristically small. This fact underscored the importance of seeking special structures within a polyhedral environment that either directly provides facetial constraints, or, as later will prove equally as important, affords access to such facetial constraints. Moreover, note that the cutting planes generated by any subroutine were appended to the *main* linear programming relaxation problem, and not simply some branch-specific subproblem, thereby insuring that they were globally valid for the entire branch-and-bound decision tree. This provides a computational advantage for a serial computing machine, but not necessarily for a parallel architecture. As discussed in Chapter 6, adaption of a scheme to generate only globally valid constraints could introduce additional complexity within a parallel implementation using distributed memory, since the amount of necessary communication between processors to pass cutting plane information might increase substantially. Thus, although pursuing globally valid cutting planes for the purposes of generating tighter initial formulations retains its importance, their pursuit during runtime might sacrifice computational efficiency.

1.5.2. Cutting planes for 0-1 problems. Crowder, Johnson and Padberg (1983) extended the general approach of Crowder and Padberg (1980) to the case of large-scale 0-1 linear programming problems, and provided further proof that cutting planes related to facets of the underlying polytope were indispensable tools for the exact solution of hard combinatorial optimization problems. Their focus was on problems of the form:

$$\text{Minimize } \{cx : \ Ax \leq b, \ x_j \in \{0,1\} \ \text{ for } j = 1, \ldots, n\}, \tag{14}$$

where A is an $m \times n$ matrix with arbitrary rational entries, and b and c are vectors of length m and n, respectively, having rational entries. The components of the cost vector c were transformed to nonnegative values. The ten problems examined in their study possessed a *sparse* matrix A, i.e., the total number of nonzero elements a_{ij} of A divided by the product mn was typically less than 0.05. Aside from sparsity, about half of the sample problems had no apparent special structure, and the remainder possessed special structures not directly exploitable by the approach that follows. It should be noted that the special structures that Crowder, et al. (1983) were looking for

were polyhedral special structures. As will be shown in the sequel, most of the problems possessed some algebraic structure that could be exploited to tighten the linear programming relaxation, if not to produce facetial cutting planes.

Several novel *ad hoc* preprocessing procedures were used to improve the associated linear programming formulation. These procedures keyed on various algebraic characteristics of the individual constraints defining (14). A given problem formulation was first preprocessed to: (a) identify variables that could be fixed at zero or one and check for blatant infeasibility, (b) tighten the constraint set through coefficient reduction, and (c) determine constraints of the problem rendered inactive by previous manipulations. These preprocessing steps have become standard routines in current mixed-integer programming and pure integer programming software packages.

For example, consider a 0-1 constraint such as

$$\sum_{j \in N_+} a_j x_j + \sum_{j \in N_-} a_j x_j \leq a_0, \tag{15}$$

where N_+ denotes the index set of coefficients a_j having positive values and N_- those with negative values. If $\sum_{j \in N_-} a_j > a_0$, then (15) does not have a feasible solution, and (14) is *blatantly infeasible*. If, on the other hand, $\sum_{j \in N_+} a_j \leq a_0$ holds, then (15) is *redundant* because every possible zero-one solution to (14) satisfies it, and it can be dropped from the problem formulation.

In general, a coefficient reduction rule for a constraint of the form (15) can be stated as: let $Q = \sum_{j=1}^{n} \max\{0, a_j\}$, then

(1) if $a_k < 0$ and $Q < a_0 - a_k$, then change a_k to $a_k' = a_0 - Q$;

(2) if $a_k > 0$ and $Q < a_0 + a_k$ then change a_k to $a_k' = Q - a_0$ and change a_0 to $a_0' = Q - a_k$.

For example, in the constraint $4x_1 - 3x_2 + 2x_3 \leq 4$, we have $Q = 6$. By applying the second rule, a_1 and a_0 can be changed to $a_1' = 2$ and $a_0' = 2$, respectively. Next, applying the first rule to a_2 changes this coefficient to $a_2' = -2$, resulting in the tightened constraint $2x_1 - 2x_2 + 2x_3 \leq 2$. Figure 3 illustrates the intersection of the final inequality (in dashed lines), which is actually facetial, with the unit hypercube.

When the sense of the inequality of (15) is reversed, and an inequality is of the form $\sum_{j \in N} a_j x_j \geq$

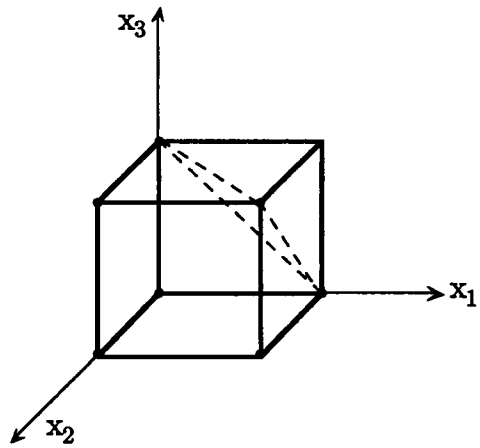

FIGURE 3. Facetial constraint by coefficient reduction.

a_0, where $a_j \geq 0$ for all $j \in N$, if $a_k > a_0$ for some $k \in N$, then a_k can be replaced by a_0, yielding

$$a_0 x_k + \sum_{j \neq k \in N} a_j x_j \geq a_0. \tag{16}$$

These tightened inequalities have the same set of 0-1 solutions as the original inequalities, but fewer real solutions. In fact, although coefficient reduction may occasionally produce facetial constraints as in the example above, it always reduces the linear programming relaxation feasible region. Since every constraint of (14) can be transformed into this form via the variable subsititution $x_j' = 1 - x_j$, every constraint was examined in this manner. Then, appropriate reductions were applied and the resulting constraint was transformed back into the original variables via a reverse substitution (if necessary) and replaced in (14). At the end of the preprocessing stage, the resulting linear programming relaxation was solved to optimality in order to obtain a valid lower bound to use in a branch-and-bound procedure, as before.

The cutting planes generated by Crowder, et al. (1983) for this type of problem again maintained global validity for the entire branching tree. The method of generating these constraints represented a union of the polyhedral research of Padberg (1973) and Balas and Zemel (1978) with computational algorithms. This technique was entirely based on implicit structures available solely through polyhedral theory. Let

$$P_I^i = \text{ conv } \{x \in R^n : \sum_{j=1}^{n} a_j^i x_j \leq b^i,\ x_j \in \{0,1\},\ j = 1, \ldots, n\} \tag{17}$$

denote the convex hull of 0-1 solutions to the single inequality $a^i x \leq b^i$, $i = 1, \ldots, m$, so that P_I^i represents the knapsack polytope associated with constraint i of (14). Similarly, let

$$P_I = \text{ conv } \{x \in R^n : Ax \leq b,\ x_j \in \{0,1\},\ j = 1, \ldots, n\} \tag{18}$$

denote the convex hull of 0-1 solutions of the entire constraint set of (14). It is clear that $P_I \subseteq \bigcap_{i=1}^{m} P_I^i$, and that equality does not, in general, hold. However, Crowder, et al. (1983) noted that in the case of large-scale 0-1 programming problems having a sparse matrix A and no apparent special structure, it is reasonable to expect that the intersection of the m knapsack polytopes P_I^i, defined by the m constraints of (14), provides a fairly good approximation to the 0-1 polytope of (14) over which the problem attempts to minimize a linear objective function. Thus, the constraints of (14) could be preprocessed as individual knapsack constraints with a reasonable expectation of tightening the polytope P_I.

The particular facetial constraints generated by the cutting plane routine exploited two *implicit* structural inequalities not incorporated into previous strategies: lifted minimal covers, and lifted $(1, k)$-configuration constraints. For a linear inequality

$$\sum_{j \in K} a_j x_j \leq a_0, \tag{19}$$

where $a_j > 0$, $j \in K$, and the variables x_j, $j \in K$, are binary, denote $S \subseteq K$ so that

$$\sum_{j \in S} a_j > a_0 \qquad \text{and} \qquad \sum_{j \in S} a_j - a_k \leq a_0, \ \text{ for all } k \in S, \tag{20}$$

holds. The index set S is then called a *minimal cover* with respect to (19), and every 0-1 solution to (19) satisfies the minimal cover inequality

$$\sum_{j \in S} x_j \leq |S| - 1, \tag{21}$$

where $|S|$ denotes the cardinality of the set S. If $K = S$, then this minimal cover constraint defines a facet of the associated knapsack polytope. Otherwise, it is subsequently lifted, as in (4), to a facetial constraint.

The second type of implicit structure exploited by Crowder, et al. (1983), $(1, k)$-*configuration* constraints, are a more general class of constraints that contains minimal covers, and result from

the following reasoning. Suppose that $S^* \subseteq K$ and $t \in K \backslash S^*$ satisfy the two conditions:

$$\sum_{j \in S^*} a_j \leq a_0, \tag{22}$$

and

$$Q \cup \{t\} \text{ is a minimal cover for every } Q \subseteq S^* \text{ with } |Q| = k, \tag{23}$$

where k is an integer such that $2 \leq k \leq |S^*|$. Because of the one index role of t and the integer k, the set $S^* \cup \{t\}$ is called a $(1, k)$-configuration with respect to (19). Then, every 0-1 solution to (19) satisfies the set of inequalities

$$(r - k + 1)x_t + \sum_{j \in T(r)} x_j \leq r, \tag{24}$$

where $T(r) \subseteq S^*$ varies over all subsets of cardinality r of S^*, and r varies over all integers from k to S^*, inclusively. If $K = S^* \cup \{t\}$, then (24) defines a facet of the associated knapsack polytope. Otherwise, it too is lifted as in the case of a minimal cover constraint. Notice that, if $k = |S^*|$ holds in the two conditions (22) and (23), then a $(1, k)$-configuration is a minimal cover.

At the time this particular approach was introduced, no technically good algorithm existed to solve the 0-1 knapsack problem required to identify the facetial versions of these two special structures having x_j for $j \in S$, or $j \in S^*$ as binary. Consequently, Crowder, et al. (1983) implemented a *continuous* version of identifying and lifting the desired minimal cover and $(1, k)$-configuration constraints within their algorithm, hence providing a tightening through *approximated* facets of the underlying polyhedron of (14). Later, Zemel (1989) demonstrated how the actual lifted facets of these constraints could be computed in $O(n|S|)$ time.

Although their assertion of the absence of special structures such as obvious GUB constraints was valid, the methodology introduced by Crowder, et al. (1983) was, in fact, exploiting implicit special structures afforded uniquely by the 0-1 problem to obtain deep cuts to support the overall branch-and-bound strategy. As opposed to earlier cutting plane techniques, these particular cuts preserved the sparsity of the constraint matrix A because the coefficient of a lifted variable is non-zero only if the variable's original coefficient, a_j, is non-zero. Thus, the support of an inequality obtained by lifting a minimal cover constraint or a $(1, k)$-configuration constraint is contained in the support of (19).

0-1 Vars	# cuts	GAP	Fraction
33	20	269.6	0.92
40	5	197.9	0.17
201	5	490.0	0.00
282	240	81543.5	0.96
291	72	3473.8	0.94
548	139	5565.1	0.99
1550	0	1.5	0.00
1939	1	14.9	0.00
2655	2	15.9	0.19
2756	326	422.9	0.98

TABLE 2. Lifted cutting plane results.

Table 2 illustrates the effectiveness of this lifted constraint generation procedure measured by the degree to which it closes the "gap" between the optimal linear programming relaxation objective function value and the optimal 0-1 objective function value. The column GAP shows the difference between these two values after preprocessing, but before constraint generation. The last column reflects the fraction of the gap closed due to the generation of lifted minimal cover cutting planes. The problem with 548 0-1 variables illustrates the effectiveness of these facetial cutting planes more than any of the others. At the end of the preprocessing stage, which reduced a considerable amount of the gap, there remained a gap much too large to permit one to expect completion of the branch-and-bound phase within a reasonable time limit. In fact, Crowder, et al. (1983) originally attempted to solve this problem without cutting planes and had to give up after letting the problem run for several CPU hours. However, after identifying and generating the cutting planes noted, the gap after preprocessing was reduced to a small fraction of the original value, permitting the problem to be solved in less than one minute of CPU time. This observation was supported by the results of several other successful runs, including the problem with 2756 0-1 variables.

The success of Crowder, et al. (1983) provided momentum to improving the techniques used to

strengthen the initial linear programming formulation relaxation prior to attempting to solve the integer optimization problem. Johnson, Kostreva, and Suhl (1985) applied a similar methodology as that of Crowder, et al. (1983) to 0-1 integer programming problems arising from large scale planning models possessing explicit specially structured constraints:

(1) Either-or constraints of the form

$$-Mw_i + \sum_{j \in N_i} a_{ij} x_j \leq b_{i0},$$
$$-Mw_i + \sum_{j \in N_i} a_{ij} x_j \geq -M + b_{i1},$$

where for each project i , w_i is the project variable, M is a sufficiently large positive constant, and x_j, $j \in N_i$, are the activity variable that affect w_i.

(2) Mutual exclusivity, or *clique* constraints of the form

$$\sum_{j \in S_k} x_j \leq 1.$$

(3) Fixed charge constraints of the form

$$\sum_{j \in T_l} x_j \leq M y_l,$$

where x_j is some activity variable, M is again a sufficiently large positive constant, and y_l is an adjoined variable called a *fixed charge variable.*

Noting similar successes due to the criticality of having a strong linear programming formulation in hand, their method subdivided into three phases: logical methods applied before linear programming, constraint generation with linear programming, and specialized branch-and-bound for solving the preprocessed integer optimization problem. Unlike Crowder, et al. (1983), they added the concept of *probing* to the preprocessing stage which was developed by Guignard, Spielberg, and Suhl (1978). In probing, a variable x_j is set either to zero or one, and the tests are then repeated on the resulting problem. If that problem turns out to be infeasible, then the original variable can be fixed in the other direction. For the largest problems, they used probing on only the most important variables.

The constraint generation technique used to create cutting planes was the same as that used by Crowder, et al. (1983). It is interesting to observe from the results shown in Table 3, that

	Pbm 1	Pbm 2	Pbm 3	Pbm 4
No. of variables	291	313	548	2756
No. of constraints	252	272	176	755
Percent gap closed (PP)	91	86	89	49
No. of cuts added	54	138	99	128
Percent gap closed (CP)	95	94	99	77
Nodes (B-and-B)	82	214	221	3322
Percent opt proven	100	100	100	98.91

TABLE 3. Preprocessing results on closing LP–IP gap.

the largest percentage of the integrality gap was closed by the preprocessing routines, with the previously reported cutting planes making marginal gains past this initial closing. This observation further reinforced the conviction of researchers that a good initial model of the problem must be carefully constructed in the sense that it affords a tight underlying linear programming representation, and that any inherent special structures must be exploited, both in the process of model formulation and algorithmic developments.

Continuing to focus on improving the procedures for preprocessing a problem formulation to reduce the size of the integrality gap, Martin and Schrage (1985) introduced a method to generate an initial set of cuts designed to tighten both pure 0-1 and mixed-integer 0-1 problems. Their method was based on two ideas: subset selection and logical coefficient reduction. The latter idea was identical to the technique introduced by Crowder, et al. (1983), i.e., a given constraint of the form

$$\sum_{j\in I} a_j x_j + \sum_{j\in F} a_j x_j + \sum_{j\in P} a_j x_j \geq b, \tag{25}$$

where $a_j \geq 0$, $x_j \in \{0,1\}$ for all $j \in I$, $x_j \geq 0$ for all $j \in P$, and $0 \leq x_j \leq 1$ for all $j \in F$, can be replaced by:

$$\sum_{j\in I} \min\{a_j, b\} x_j + \sum_{j\in F} a_j x_j + \sum_{j\in P} a_j x_j \geq b. \tag{26}$$

However, their idea of subset selection was new. It was intended to exploit implied constraints of (25) to generate coefficient reduction cuts when (25) could not do so directly. For example, consider

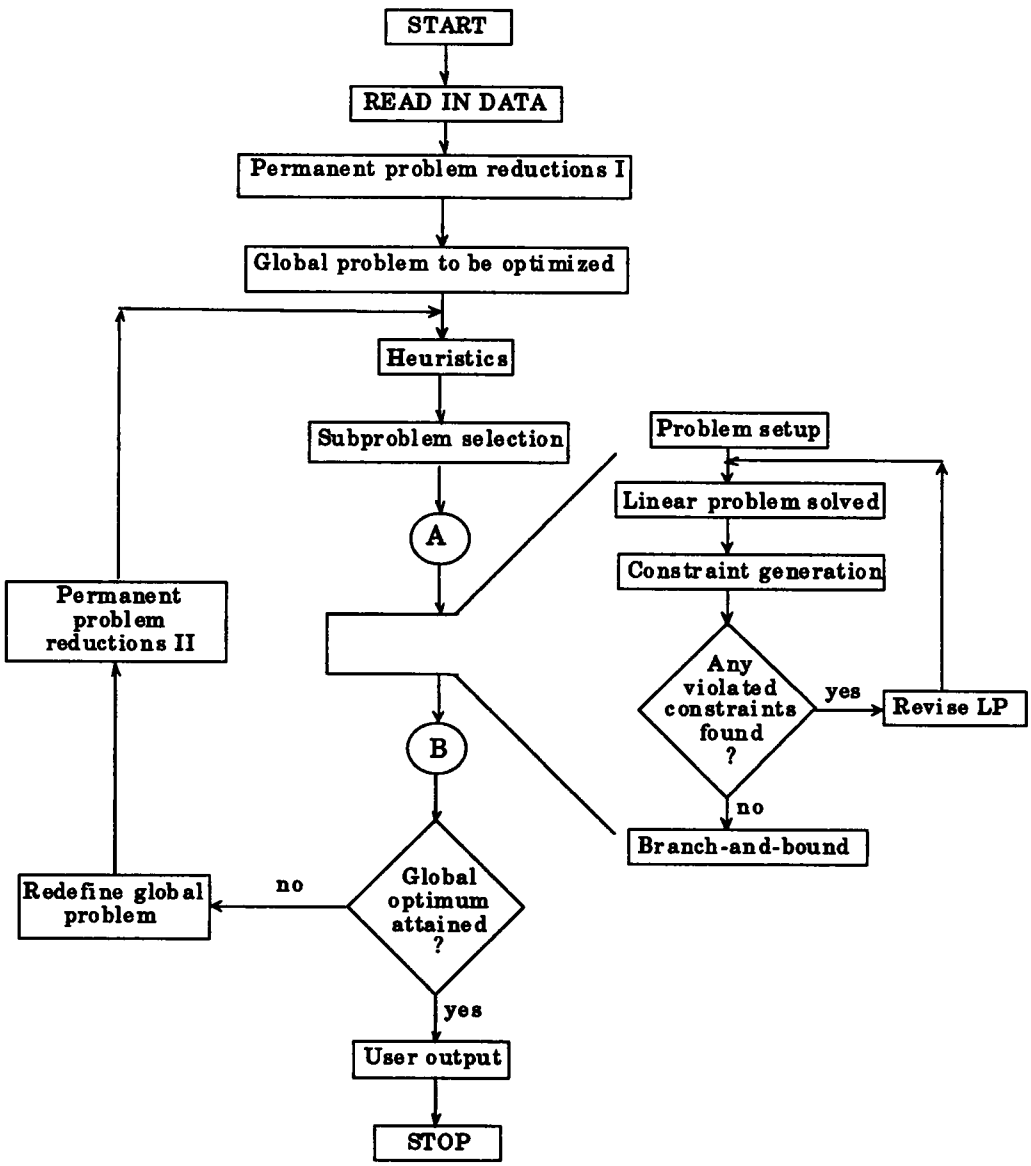

FIGURE 4. LP-based combinatorial optimization procedure.

the constraint $x_1 + x_2 + x_3 + 3x_4 + x_5 \geq 3$ where $0 \leq x_1, x_2, x_3 \leq 1$, and $x_4, x_5 \in \{0, 1\}$. Suppose that the current linear programming relaxation solution is given by $\bar{x}_1 = 10/24$, $\bar{x}_2 = 11/24$, $\bar{x}_3 = 12/24$, $\bar{x}_4 = 5/24$, and $\bar{x}_5 = 1$. Notice that standard logical coefficient reduction has no effect on this constraint. However, any feasible solution to this constraint must also satisfy the implied constraint $x_1 + x_2 + x_3 + 3x_4 \geq 2$ because the maximum value of x_5 is one. Identifying appropriate subsets of the variables in I and F to fix at a value of one, produces an implied constraint that can be subsequently tightened via coefficient reduction. In this case, the valid cut produced is given by $x_1 + x_2 + x_3 + 2x_4 \geq 2$, which cuts off the current linear programming relaxation solution.

Formally, if $K \subseteq I$ and $S \subseteq F$ and $b_0 = b - \sum_{j \in K} a_j - \sum_{j \in S} a_j > 0$, then a valid cut implied by constraint (25) is:

$$\sum_{j \in I-K} \min\{b_0, a_j\} x_j + \sum_{j \in F-S} a_j x_j + \sum_{j \in P} a_j x_j \geq b_0. \tag{27}$$

Since a problem of realistic size would preclude the practice of coefficient reduction on every possible subset, they advocated a method of selecting subsets K and S in such a way to minimize

$$\sum_{j \in I-K} \min\{b_0, a_j\} \bar{x}_j + \sum_{j \in F-S} a_j \bar{x}_j - b_0, \tag{28}$$

so that, instead of identifying the subsets K and S that satisfy $b_0 = b - \sum_{j \in K} a_j - \sum_{j \in S} a_j$ as a strict equality, they allow the minimization to identify K and S based on $b_0 \leq b - \sum_{j \in K} a_j - \sum_{j \in S} a_j$. Additionally, they again endorsed the use of lifted constraints and minimal cover cuts within any computational approach used in solving these types of problems.

No single research emphasized this point more than the seminal paper by Hoffman and Padberg (1985) that addresses linear programming based combinatorial problem solving. The fact that both pure and mixed-integer 0-1 programming problems could now be solved to optimality in economically feasible computation times by methods directly extracted from polyhedral theory validated the use of both facetial cutting planes and effective preprocessing. Since there existed only a few instances where the naive linear programming relaxation of a given integer optimization problem had been proven to solve the combinatorial optimization problem for all possible choices of the objective function, in particular, when the matrix A is totally unimodular, it was becom-

ing standard practice to tighten the linear programming relaxation as much as possible at the onset. Even though complete descriptions of the underlying polytope, P_I, for each combinatorial optimization problem were (and are still) not available, a *partial* list of facet defining inequalities based on explicit or implicit special structures present in the problem formulation significantly improved solution times. By prescribing the general methodology shown in Figure 4, the challenge of combinatorial optimization was clear: identify a sufficiently large subsystem $H^*x \leq h^*$ of the complete system $Hx \leq h$ of the polytope P_I that permits one to reduce the integrality gap as much as possible before handing the problem over to a branch-and-bound procedure.

Padberg and Rinaldi (1987) successfully implemented a slight modification to this generic approach (Figure 4) to solve to optimality a 532-city symmetric TSP involving the optimization over 141,246 zero-one variables. The cities were located as data pairs of (pseudo-Euclidean) (x, y)-coordinates of 532 cities located in the continental United States. The constraints generated for use as cutting planes consisted only of subtour elimination constraints and comb constraints, similar to those used by Crowder and Padberg (1980). However, Padberg and Rinaldi (1987) used a different manner of generating these cuts which was more closely tied to branch-and-bound. They developed a '*branch-and-cut*' scheme where, like in branch-and-bound, they selected a variable x, say, by some reasonable criterion to branch on. Enforcing the constraint $x \geq 1$ on the 'up-branch', and $x \leq 0$ on the 'down-branch', their routine moved from the current vertex to a different one which was then fed into a cut generator that finds new constraints to cut off the new vertex. The constraints generated at any given node of the search tree were again globally valid at any other node of the tree because they defined facets of the entire TSP polytope. In following such a constraint generation scheme, it was no longer necessary to keep track of the order in which constraints were generated, which was necessary for a standard branch-and-bound approach that used subproblems to generate cutting planes. Furthermore, the book-keeping could be kept to a minimum since it was no longer necessary to keep track of previous linear programming basis information and the like, because at every branching step, they did not just impose simple constraints of the type $x \geq 1$ and $x \leq 0$, but rather generated any number of violated facet-defining constraints repeatedly, so long as progress towards optimality remained satisfactory.

Capitalizing on the apparent success of this approach, researchers extended its application to

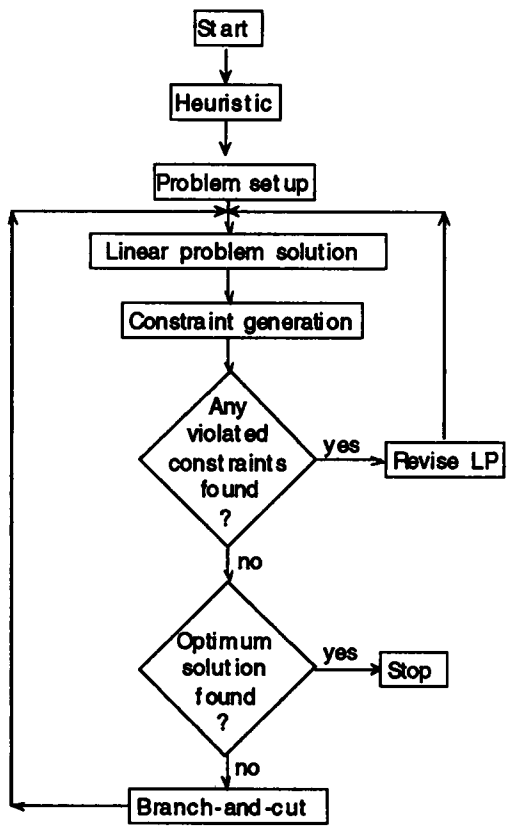

FIGURE 5. TSP branch-and-cut.

several other classes of problems in the ensuing years. Van Roy and Wolsey (1987) extended the practice of automatically reformulating a given combinatorial optimization problem via preprocessing to mixed-integer programming, noting that reformulation using strong valid inequalities was necessary for solving or finding good optimal solutions. They also suggested that this technique could easily be improved to handle generalized upper bound constraints. Goemans (1989) did exactly that, developing a family of facet-defining valid inequalities for the class of 0-1 mixed-integer programming problems with a single knapsack constraint and variable upper bounds on the continuous variables of the problem. Within this same category, Dietrich and Escudero (1990) extended the notion of coefficient reduction for 0-1 knapsack constraints in the presence of variable bounding constraints. They improved the reduction idea for this class, demonstrating that stronger reductions can be obtained by exploiting the bounding relationship between variables, an aspect not considered by Crowder, et al. (1983). Later, Dietrich and Escudero (1992) applied a similar coefficient strategy to tighten cover induced inequalities using 0-1 knapsack constraints, and, if available, cliques whose variables were included in the cover. In this case, tightening is achieved by increasing the coefficients of the cover inequality, since they are generally of the form $\sum_{j \in J} c_j x_j \leq k$, where $c_j \in \{0, 1\}$ gives the cover coefficient for j, and where $0 < k < \sum_{j \in J} c_j$, $k \in Z^+$. Note that if $k = 1$, then this cover induced inequality corresponds to a clique inequality.

Padberg and Rinaldi (1991) provided a more in-depth treatment of branch-and-cut for large-scale symmetric traveling salesman problems, noting, among other results, the '*tailing-off*' effect on the best LP relaxation objective function value, $v(LP)$, obtained at node 0 via cut generation. For example, in one particular problem involving 263 cities, the value of $v(LP)$ increases quickly in the first 10 iterations, then it increases by less than 9 units in the following 30 iterations, and finally it takes 71 iterations to increase by less than half a unit. This '*tailing off*' effect reveals the inability of the cut generator to produce the 'right' cut that would take the current optimal LP solution out of the corner of the polytope where it is trapped. They observed that the stronger the cut generator, the higher is the value of $v(LP)$ where the tailing off occurs. A very tight initial linear programming formulation reduces the size of the integrality gap where tailing can occur, and has the potential to render this effect insignificant, independent of the quality of subsequently generated cuts. This fact alone supports efforts to provide as tight a formulation as

possible from the onset. (See Hoffman and Padberg (1991) for a very comprehensive survey of the various techniques for automatically improving the linear programming representation of general 0-1 linear programming problems.)

For some specially structured problems, a variable redefinition technique had also been proposed by Martin (1987) for mixed-integer linear programming problems of the form

(EM) Minimize cx

subject to: $Ax \geq a$,

$$x \in X = \{x : Bx \geq b,\ x \geq 0,\ x_j \text{ integer for all } j \in J\},$$

where the constraints $Ax \geq a$ represent the complicating or coupling constraints, and the $Bx \geq b$ constraints represent constraints containing some identifiable special structure. Whereas the efforts of Crowder and Padberg (1980), Crowder, et al. (1983), and Hoffman and Padberg (1983) strived to close the integrality gap by characterizing either fully or partially the facial structure of $\text{conv}(X)$, the foundation of this approach rested on the idea that by dropping certain complicating constraints, a specially structured subproblem would result that could be reformulated using an entirely different set of decision variables, or a subset of the original x variables plus some new auxiliary variables. This reformulation process yields an equivalent problem formulation that tightens the linear programming relaxation by constructing a partial convex hull for a subset of constraints. The selection of a linear transformation to relate different mixed-integer formulations is a natural one because the image of a polytope under a linear transformation is also a polytope.

Letting Z denote the feasible region in this new variable space that corresponds to X, the entire problem (EM) could be reformulated using an appropriate linear transformation T so that the following two conditions held:

(1) every extreme point of $\text{conv}(X)$ is contained in the image of Z under T; and,

(2) $T(Z) \subseteq \text{conv}\ (X)$.

The stronger requirement that $T(Z) \subseteq X$ was additionally required so that an optimal solution z^* to $\min\{cTz : z \in Z\}$ implies that $x^* = Tz^*$ was an optimal solution to $\min\{cx : x \in X\}$.

Eppen and Martin (1987) reported encouraging results applying this methodology to solving multi-item capacitated lot-sizing problems. In five out of the seven problems tested, the optimal

linear programming relaxation solution was identical to the optimal solution to the integer programming problem. For the other two problems, the integrality gap was closed to 1.9% and 0.15% of integer optimality.

Martin (1991) introduced a new method of automatically generating auxiliary variable reformulations using a cutting plane method for problems of the type

$$\text{Minimize } \{cx : Dx \geq d,\ x \in P \subseteq R^n\}, \tag{29}$$

where P is the polyhedron given by

$$P \equiv \{x : a^i x \leq a_0^i,\ i = 1, \ldots, q,\ x \geq 0,\ x_j \leq f_j,\ \forall j \in J\}.$$

The coefficients c and a^i are n-vectors, and a_0^i and f_j are scalars. The constraints $a^i x \leq a_0^i$ can be interpreted as valid cuts for the mixed integer program defined by requiring the variables indexed by the set J to be integers.

Consider the linear program

$$\max\{xa + g\theta : Fa + H\theta \leq 0,\ \theta \geq 0\}, \tag{30}$$

where a and θ are column vectors of real variables conformable with the row vectors of real numbers x and g respectively, and the matrices F and H. Then, (30) is a valid *separation program* for P if given $\bar{x} \geq 0$, $\bar{x}_j \leq f_j$ for all $j \in J$, the optimal value of (30) equals zero if and only if $a^i \bar{x} \leq a_0^i$, $i = 1, \ldots, q$. Given the existence of a valid separation linear program, the problem (29) can be reformulated using auxiliary variables z as:

$$\begin{aligned} &\text{Minimize } cx \\ &\text{subject to: } Dx \geq d, \\ &\qquad x = zF, \\ &\qquad zH \geq g, \\ &\qquad x_j \leq f_j \qquad \forall j \in J, \\ &\qquad x, z \geq 0. \end{aligned}$$

Although encouraging computational results were obtained for problems in which this separation problem was embedded in a cutting plane algorithm, perhaps equally as interesting, was the

open research question posed by Martin motivated by the insights provided by this reformulation strategy: can a polynomial size reformulation be generated in this manner for any problem that has a polynomial separation algorithm for finding valid cuts?

Permeating throughout all these developments is a sense that perhaps there might exist some unifying theory that, taken as a whole, would either identify commonly shared characteristics of the underlying polytopes, or define a hierarchical structure to tightening linear programming relaxations afforded by the various classes of facetial and non-facetial deep cuts. This goal differs significantly from simply a generalization of the various techniques, such as that proposed by Dietrich, Escudero, and Chance (1993) for 0-1 optimization problems.

Their methodology provided two generalizations of "ideal" reformulations that addressed individual knapsack constraints, pairs of knapsack constraints, clique and cover induced inequalities, variable upper bounding constraints and capacity expansion constraints. However, the missing component in this, and other approaches of a similar nature, was the specification of an entire hierarchy based upon a single approach that has the potential to recover an entire convex hull representation, if one was willing to expend a sufficient amount of algebraic and computational effort. Ironically, such a unifying hierarchy was always obtainable, but was effectively concealed behind the idea that a combinatorial optimization problem must first be moved into a higher dimensional space, transferring integer complexity to nonlinear complexity. By subsequently relinearizing the problem, and projecting it back into the space of original variables, any desired degree of tightening could then be achieved. This methodology provides the nucleus of a Reformulation-Linearization Techniques (RLT) whose theory underlies the developments of this dissertation.

1.5.3. Reformulation-Linearization Technique (RLT). The fundamental RLT procedure introduced by Sherali and Adams (1989, 1990, 1994) begins by first reformulating a given problem. This is done by defining a set of nonnegative factors using the problem variables' bound factor constraints, and suitably multiplying combinations of these factors with the original problem constraints to generate additional valid nonlinear constraints. These resulting constraints are then subsequently linearized by defining a new set of variables, one for each nonlinear term. Depending on the specifics of its application, and hence the structure of the problem, this RLT process yields various linear programming representations or relaxations that have specific, de-

sirable characteristics. In particular, the addition of auxiliary variables in RLT strengthens the linear programming formulation in such a way that it makes solution by branch-and-bound either significantly easier, or altogether unnecessary.

In the next chapter, we present a comprehensive literature review of the development of this technique along with several variations that are currently under active development. Here, we briefly mention several classes of problems that have succumbed to the Sherali-Adams RLT in recent years in order to illustrate the extent of its applicability to discrete and nonlinear optimization problems.

RLT was originally designed to employ factors involving 0-1 variables stemming from problems of the form

$$\text{Minimize } \{cx + dy : (x, y) \in X,\ x \text{ binary },\ 0 \leq y \leq u\},$$

in order to generate the required nonlinear (polynomial) reformulation. For these types of 0-1 mixed-integer linear, and also extendable to 0-1 polynomial (multilinear) programming problems, Sherali and Adams (1989, 1990, 1994) have shown that such an RLT can produce a hierarchy of tighter and tighter linear programming relaxations, recovering the convex hull of integer feasible solutions at the highest level of relaxation.

In another study, Balas, Ceria and Cornuejols (1993) described encouraging computational results in testing a cutting plane algorithm called a Lift-and-Project algorithm that incorporates cuts based upon first level RLT relaxations. More recently, Balas, Ceria and Cornuéjols (1994) have applied this cutting plane approach to the maximum clique problem, demonstrating a high degree of closure of the integrality gap when using cuts based on the special structure afforded by the clique constraints.

A related hierarchy of relaxations was independently developed by Lovasz and Schrijver (1991) which amounts to deriving a first level tightening X_1 over the linear programming relaxation X_0, finding the projection X_{P1}, and then repeating this step by replacing X_0 with X_1. Continuing in this fashion, they demonstrated that in n steps, the convex hull of X is obtained. Their requirement of interspersing a projection operation, however, renders their procedure as impractical for computational purposes.

Sherali and Tuncbilek (1992) adapted the RLT approach of Sherali and Adams (1989) in concert with a suitable partitioning technique to develop an algorithm that was proven to converge to a global optimum for *continuous*, nonconvex, polynomial programming problems of the form

$$\text{Minimize } \{\phi_0(x) : \phi_r(x) \geq \beta_r \text{ for } r = 1, \ldots, R_1, \ \phi_r(x) = \beta_r \text{ for } r = R_1 + 1, \ldots, R, \ x \in \Omega\},$$

where

$$\phi_r(x) \equiv \sum_{t \in T_r} \alpha_{rt} \left(\prod_{j \in J_{rt}} x_j^{\alpha_{jrt}} \right) \text{ for } r = 0, 1, \ldots, R,$$

and where

$$\Omega \equiv \{x : 0 \leq l_j \leq x_j \leq u_j < \infty, \text{ for } j = 1, \ldots, n\}.$$

Sherali and Tuncbilek (1992) also developed a suitable application of the RLT concept for the location-allocation problem in which the transportion cost penalty is proportional to the squared Euclidean distance between the supply and destination points:

$$\text{Minimize } \{\sum_{i=1}^{m} \sum_{j=1}^{n} w_{ij} \left((x_i - a_j)^2 + (y_i - b_j)^2 \right) \ : \ w \in W\}.$$

Projecting this problem into the space of w-variables alone, and expressing the result as a minimization of a concave quadratic function over the transportation constraints, they applied RLT to this w-space problem to generate a tight linear programming relaxation via a piecewise linear convex underestimator of the objective function. Furthermore, they were able to exploit the structure of the transportation constraints by devising suitable logical tests, and generating cut-set inequalities based on a cycle prevention method that preserves a forest graph for any partial solution to further tighten the lower bound. In computational testing, the initial linear program itself produces solutions within 2–4% of optimality. This enabled them to solve problems having up to 20 facilities and 120 customers, while previous algorithms were unable to solve 4 facility and 10 customer problems within reasonable effort.

Using only low-level relaxations created by the RLT procedure, Sherali and Alameddine (1990) derived an explicit representation of the convex envelope of the nonconvex bilinear function over

so-called D-polytopes for the problem

$$\text{(BLP)} \qquad \text{Minimize } \phi(x,y) \equiv c^T x + d^T y + x^T G y$$

$$(31) \qquad \text{subject to: } (x,y) \in Z \cap \Omega,$$

where $x \in R^n$, $y \in R^m$, Z is a polyhedron in R^{n+m}, and Ω is a hyperrectangle of the form $\Omega \equiv \{(x,y) : 0 \leq l \leq x \leq u < \infty, 0 \leq L \leq y \leq U < \infty\}$. They showed that when $Z \cap \Omega$ is a triangular or quadrilateral D-polytope in R^2, the actual closure convex hull of $\overline{Z}$ is obtained, which can be useful in a branch-and-bound framework. In the more general case, an RLT procedure was developed by Sherali and Alameddine (1992) that yielded a useful lower bounding linear program. This was embedded within a convergent partitioning procedure that was demonstrated to significantly outperform existing techniques based on convex envelopes. For almost all separably constrained problems of this type, and an overwhelming number of jointly constrained problems of this type, they reported that the initial linear programming relaxation itself solved the bilinear problem. When this was not the case, the initial gap between the lower and upper bounds was close enough to produce an optimum after the branch-and-bound algorithm enumerated only a few nodes. However, it can be the case, as they note, that the size of the bounding linear program created by a complete RLT application can be prohibitively large. In light of this, they proposed a 'constraint filtering strategy' that might be implemented to dispense with constraints that might not contribute significantly to the tightness of the resulting bounds. Briefly, assuming that the constraints of a particular linearization $LP(\Omega)$ are generated in the usual RLT manner, for each inequality constraint $\alpha_i z \geq \beta_i$, say, a signed Euclidean distance of $\bar{z}$ can be computed from the corresponding hyperplane as $(\alpha_i \bar{z} - \beta_i)/\|\alpha^i\|$. If the distance exceeds a specific tolerance $\tau > 0$, the constraint is deleted. The resulting reduced problem is then used to compute a lower bound.

Although the several classes of optimization problems mentioned appear to have succumbed to improved algorithms, clever reformulation techniques, and advances in computing technologies, the class of 0-1 mixed-integer programs (MIP) in discrete optimization remain steadfast in their challenge. Despite the fact that individual cases have been studied from the perspective of tightening linear programming relaxations and exploiting special structures for which success has been noted, there has not existed to date a unifying framework that implements both of these

tasks within an automatic facility. This study introduces a general technique for constructing tight model reformulations while exploiting special structures, based upon the Reformulation–Linearization Technique (RLT) of Sherali and Adams (1989, 1990, 1994).

Specifically, it is shown how certain inherent special structures present in a problem formulation could be exploited to develop specific classes of multiplicative factors that can be applied to the constraints of the original problem in order to reformulate it as an equivalent polynomial programming problem. This polynomial programming problem can then be subsequently linearized to produce a tighter relaxation in a higher dimensional space. This general framework permits one to generate a hierarchical sequence of tighter relaxations, leading up to the convex hull representation. Because of the focus on developing multiplicative factors based on special structures, an efficiency is achieved that has generally proved to be elusive in previous reformulation-linearization methods. It is possible, for several classes of combinatorial optimization problems, to *a priori* identify the nature of the partial convex hull representation that might result for a particular level of algebraic and computational effort. The potential savings that could be realized for an enumerative algorithm such as branch-and-bound is significant.

1.6. Parallel processing

There has been an increasing amount of research performed in recent years focussing on adapting the branch-and-bound algorithm to take advantage of parallel architectures of computing machines involving more than a single processor (see Gendrion and Crainic (1994)). In addition to differences in hardware design, the method of controlling operations in each environment differentiates implementation approaches. The main modes of computation can be grouped as *control driven*, *data driven*, and *demand driven* (Treleaven, Brownbridge and Hopkins (1982)). Control driven computations rely on the user to specify the exact type and order of operations to be performed. In a data driven model, operations can be performed as soon as all the necessary operands are available. The demand driven model waits to perform an operation until some required outcome is needed. All sequential computers essentially use the control driven method, whereas parallel machines appear to operate most effectively in a data driven, or *data flow* environment.

1.6.1. Parallel aspects of branch-and-bound. The branch-and-bound process contains elements that are amenable to both *course grained* parallelism, and *fine grained* parallelism, each of which can be employed when executing the algorithm. Course grained parallelism occurs when a program contains certain statements that can be executed in parallel (*e.g.*, a FORTRAN FOR loop). A sequential list of statements that are independent of one another is an example of fine grained parallelism. Whether one type of parallelism is preferred over the other is dependent upon the specific configuration of memory modules and processors.

As Gendrion and Crainic (1994) note, the development of parallel branch-and-bound research followed a chronology of early experiments (1975–1982), theoretical studies (1983–1986), and experiments on actual parallel systems (since 1987), with many contributions being made in the process. It is interesting to note that the first use of parallel processing to solve a combinatorial optimization problem by branch-and-bound is credited to Pruul (1975). Simulating a shared memory system with p processors, $1 \leq p \leq 5$, because of the lack of parallel hardware, Pruul (1975) nonetheless applied the new methodology to ten 25-city ATSP problems incorporating a bounding mechanism based on the assignment problem, subtour elimination branching rules, and a parallel depth-first branching rule. The results of Pruul's study were of little practical value at the time, though this conclusion is clearly not the case today.

Three salient aspects of the branch-and-bound algorithm lend themselves to parallelism. The first introduces parallelism in the process of performing operations on subproblems generated via branching. For example, after a number of subproblems are generated at the onset, the independent processors could be used to evaluate both the lower bounds available via the subproblems' linear programming relaxations and the best upper bounds available for each subproblem. The ideal number of subproblems to generate are as many as would be sufficient to avoid *processor starvation* due to a small number of available tasks, which typically hampers speed-up in the early stages of execution.

It is also possible to build the branch-and-bound binary search tree in parallel by performing operations on several subproblems simultaneously. Of course, this implies that each of the processors must have a sufficient capability to execute all the necessary operations of branch-and-bound independently, as in a so-called course grained MIMD (multiple instruction, multiple data) system.

Using such a scheme, Boehning, Butler and Gillett (1988) introduced a parallel branch-and-bound algorithm for integer linear programming problems that was able to achieve superlinear speedup for a subset of the test problems. Each parallel processor selected a subproblem to work on from an available pool of problems shared by one or more processors. The same basic sequence of instructions were executed by each processor: (a) request a problem, (b) add a down row, (c) add an up row, (d) analyze the down node, (e) check for fathoming, (f) put the down node in the pool of subproblems, (g) analyze the up node, (h) check for fathoming, (i) put the up node in the pool of subproblems, and repeat. Their algorithm also made use of cutting planes parallel to the objective function to cut off fractional linear programming solutions obtained for the different node subproblems.

Miller and Pekny (1989) and Pekny and Miller (1992) implemented a version of branch-and-bound based on a *processor shop model* for the asymmetric traveling salesman problem in which processors examine the list of tasks available and choose one based on some priority rules. They then process the chosen task, place the results in the correct memory location, and extract another task. Within this environment, problems ranging from 50 to 3000 cities were successfully solved on a 14 processor BBN Butterfly Plus computer. Kudva and Pekny (1993) experienced similar successes testing this approach using various sized instances of the multiple resource constrained sequencing problem. All of the implementations noted achieved success relying on course grained MIMD systems. Although this aspect of branch-and-bound has received some attention on fine grained SIMD (single instruction, multiple data) systems (Kindervater and Trienekens (1988)), difficulties were encountered trying to effectively use the SIMD architecture – a mismatch of algorithms with hardware.

The last aspect of parallelism for branch-and-bound algorithms arises in the ability to construct different branching trees in parallel by performing operations on several subproblems simultaneously. This type of decomposition is reminiscent of performing sensitivity analysis in linear programming for simultaneous variations in right-hand side values. Each of the branching trees execute different branching, bounding, and evaluation rules, and the information generated in one tree can possibly contribute to the construction of another. Miller and Pekny (1993) implemented this strategy by varying only the branching rules. Kumar and Kanal (1984) allowed each of the

processors to execute the usual lower bounding technique, but implemented an upper bound strategy that optimistically diminished each processor's known best solution by some amount from the incumbent solution stored on a single processor. Convergence of the process was thus guaranteed since the true integer solution could, at last resort, be found in finite time on the branch maintaining the actual best upper bound. Lastly, Janakiram, Gehringer, Agrawal and Mehrotra (1988) experimented with adding a stochastic character to the algorithm by randomizing the selection of the next subproblem to be evaluated by each processor. Their efforts were motivated by the reasoning that the mapping of randomized algorithms onto multiprocessors involves very little scheduling or communications overhead. To avoid possible duplication of work by processors, their technique maintained a global listing of the status of the subproblems at the first k levels of the branching tree.

Recognizably, all of the implementations of branch-and-bound strategies are dependent upon the type of parallel architecture used by the computing device, a point explained indepth in Chapter 6. Independent of this differentiation, results from the studies noted are very encouraging, and efforts into this particular vein of research for attacking hard combinatorial optimization problems are expected to blossom in the future. This will especially become the case if the current trend of increased availablity of multiprocessor desktop computers continues, along with an accompanying decrease in cost.

1.7. Organization of dissertation

The remainder of this dissertation is organized as follows. Chapter 2 presents a comprehensive literature review of the relevant linearization and reformulation-linearization techniques leading up to the hierarchy of relaxations for specially structured problems. Although several other researchers have proposed reformulation-linearization techniques that are variations of the original Sherali-Adams technique, they remain, for the most part, existential in nature because of the difficulty in providing explicit algebraic representations that can be subsequently used to derive other valid cuts. These techniques are also presented for completeness.

Chapter 3 formally introduces the new special-structure-RLT, called **RLT1**, that provides a unifying framework for constructing a spectrum of continuous relaxations, spanning from the linear

programming relaxation to the convex hull representation for linear and polynomial mixed-integer 0-1 problems. In addition to presenting the relevant theory, we demonstrate that whenever special structures are present, there is a general unifying framework that one can rely on to generate tight and often reduced sized relaxations. Moreover, in addition to explicitly recognizing existing special structures in developing specific factors in the RLT process, this hierarchy also subsumes the Sherali-Adams hierarchy as a special case.

We also illuminate the fundamental aspects of this theory in Chapter 3, presenting several examples for various special structures in order to demonstrate how these underlying special structures, including GUB, VUB, covering, partitioning and packing constraints, as well as sparsity, can be exploited within this framework. For some types of structures, we demonstrate how low-level relaxations can to recover the convex hull of integer feasible solutions. We also present an alternative partial application of this new hierarchy, along with some additional cases that lend themselves to such a scheme.

Based on the concepts of this special-structure-RLT, we introduce a new technique of tightening the inequalities generated by **RLT1** via the conditional use of logic. We show that for the class of knapsack inequalities, specific logically tightened **RLT1** inequalities recover those obtainable via a sequential lifting process. In fact, we prove that the valid inequality generated at any stage of this sequential lifting process is implied by the sequentially generated **RLT1** constraints in which the first level products are applied to the previous stage's lifted constraints. Within this context, the sequentially lifted minimal cover constraints introduced by Balas and Zemel (1978) and demonstrated by Crowder, et al. (1983) and Martin and Schrage (1985) to be so successful in tightening the bounds of linear programming relaxations, are subsumed by **RLT1**.

In Chapter 4, we apply this technique in detail to several formulations of the asymmetric traveling salesman problem (ATSP), introducing a tightened version of this formulation using Miller, Tucker, and Zemlin (1960) subtour elimination constraints. Additionally, a new formulation for the 3-index ATSP is proposed based on the quadratic assignment formulation of the ATSP. The motivation for applying **RLT1** to this arena is that the ATSP is an important and classical combinatorial optimization problem that happens to possess rich exploitable special structures. Moreover, many exact or heuristic techniques that have been developed for the solution of hard

combinatorial optimization problems were originally conceived and tested on the TSP, which, as a result, has become a prototype problem in this sense.

In Chapter 5, we then demonstrate the relative strengths of the special-structure-RLT relaxations via two computational experiments. One experiment centers on a comparison of effectiveness between **RLT1** and **RLT0** in the context of the set packing problem. Several results are presented that provide unique insights into the algorithmic nature of **RLT1**, and suggest ideal conditions under which to apply the new hierarchy. The second experiment is designed to ascertain the relative strengths of the five formulations of the ATSP introduced in Chapter 4, as the underlying cost structure is varied. Throughout, we provide several recommendations for incorporating constraints generated by **RLT1** within a branch-and-cut/bound strategy of the type introduced by Crowder and Padberg (1980), Padberg and Rinaldi (1991), Crowder, et al. (1983), and Hoffman and Padberg (1983). Lastly, we conclude this dissertation in Chapter 6 with comments concerning follow-on research of **RLT1**, and considerations toward extending this new hierarchy into a parallel computing environment.

CHAPTER 2

Linearization and Reformulation-Linearization Techniques

The Sherali–Adams Reformulation-Linearization Technique (RLT), summarized formally in this chapter, generates a hierarchy of relaxations for 0-1 mixed-integer problems (MIP), linear or polynomial, that has as many levels as the number of 0-1 variables, with the zeroeth level relaxation being the ordinary continuous relaxation, and the nth level relaxation affording the convex hull representation. The relaxation at any level in this hierarchy is generated by first reformulating the given problem as an equivalent polynomial program via a process of multiplying the constraints with certain polynomial factors composed of the bound factors $x_i \geq 0$ and $(1-x_i) \geq 0, i = 1, \ldots, n$, where $x_1, \ldots, x_n$ are the 0-1 variables. (Additional factors composed from the other defining constraints can be used in a likewise fashion in order to construct a tighter relaxation at any given level, although this is not necessary to recover the convex hull representation at level n.) Following this, the problem is linearized by defining a new variable for each polynomial term, hence the name RLT. Elements of the first level construction in this hierarchy appeared earlier in Adams and Sherali (1990), and the computational effectiveness of such partial first level constructions was exhibited in Adams and Sherali (1986, 1993). Using a *multi-homogeneous* homotopy approach, Watson and Morgan (1992) have shown that the necessary optimality conditions for the continuous polynomial programming problem can be formulated as a polynomial system of equations, among whose zeros the global optimum must lie. This methodology was demonstrated to be very practical for small problems, and has found application in geometric modeling and prototype structural design. Both the mixed integer version, and the continuous version of the polynomial programming problem over larger sized applications remain a challange.

2.1. Early linearization strategies

Some of the underlying linearization concepts of the Sherali–Adams RLT procedure, although not stemming from reformulation ideas, appeared earlier in the context of polynomial 0-1 programs. One such effort was introduced by Fortet (1959, 1960). He proposed a linearization technique for 0-1 polynomial programming problems that replaces each polynomial term with a single additional 0-1 variable and two additional constraints. Every distinct product of the form $\prod_{j \in H_h} x_j$ comprised purely of 0-1 variables is replaced by a new 0-1 variable x_{n+h}. The two new constraints

$$-\sum_{j \in H_h} x_j + x_{n+h} + |H_h| \geq 1,$$

and

$$\sum_{j \in H_h} x_j - |H_h| x_{n+h} \geq 0,$$

must be added in order to ensure that $x_{n+h} = 1$ if and only if $\prod_{j \in H_h} x_j = 1$. The downside to this linearization technique is that numbers of new variables and of new constraints so introduced may be high, even for small nonlinear 0-1 programs.

Another linearization technique, attributed to Zangwill (1965) and Watters (1967), retained the integral character of the original 0-1 polynomial programming problem by replacing each polynomial term $\prod_{j \in Q} x_j$ with a single integer variable $x_Q \in \{0, 1\}$, and adding an appropriate set of constraints to the formulation that would enforce the 0-1 requirement on any solution. Given a 0-1 polynomial problem of the form

$$\begin{aligned} &\text{minimize } f(x) \\ &\text{subject to: } g_i(x) \leq 0, \qquad i = 1, \ldots, m, \end{aligned}$$

where $x = (x_1, x_2, \ldots, x_n)$ is a vector of 0-1 variables, and $f(x)$ and $g_i(x)$ are polynomials in $x = (x_1, x_2, \ldots, x_n)$, linearization is accomplished in the following manner.

Zangwill and Watters' Linearization

- Replace each term x_j^k by x_j.

- Replace each cross-product term $\prod_{j \in Q} x_j$ by the new variable x_Q that is to satisfy the constraints

$$\sum_{j \in Q} x_j - x_Q \leq q - 1,$$

$$-\sum_{j \in Q} x_j + q x_Q \leq 0,$$

$$x_Q \in \{0, 1\},$$

where q denotes the number of elements in Q. A computational study by Taha (1970) indicated that neither an application of nonlinear algorithms to the original formulation, nor the 0-1 linear transformation of the problem via this linearization technique, afforded any clear-cut advantage over the other. One performed better than the other in some situations, and vice-versa. However, the most salient limitation of this approach is its inability to handle polynomial expressions involving both integer and continuous variables, so-called mixed-integer expressions.

Petersen (1971) and Glover and Woolsey (1974) extended the linearization technique of Zangwill (1965) and Watters (1967). Their methodology represents a significant contribution to linearization strategies for several reasons. First, they proposed several algebraic rules that make it possible to replace a polynomial cross-product term involving binary variables with a *continuous* variable, rather than an integer binary variable, in contrast to previous practice. Letting $0 \leq x_Q \leq 1$, and replacing (2.1) with the constraints

$$x_j \geq x_Q \qquad \forall j \in Q, \tag{32}$$

guarantees that x_Q is automatically 0-1 when the original variables are binary.

Second, they introduced a certain economy in their linearization by further replacing (32) with substantially fewer constraints than those generated via (32) in the following manner. Letting S_j denote the set of all sets Q that contain the index j, then, for each j that appears in some set Q, (32) can be replaced by

$$|S_j| x_j \geq \sum_{Q \in S_j} x_Q. \tag{33}$$

The number of additional constraints added to the linearization then becomes only as many as there are variables appearing in cross-product terms.

Lastly, since the difficulty of pure integer and mixed-integer programming problems is dependent on the number of integer variables more strongly than the number of continuous variables, such a replacement is computationally advantageous.

Glover (1975) extended the linearization of both Peterson (1971) and Glover and Woolsey (1974) to accommodate cross-product terms involving both continous and integer variables, without the need for additional binary variables, an important precursor to the development of RLT for mixed-integer 0-1 problems. The motivation for this extension is based on the following observation. Consider a variable $w \in R^1$ and a 0-1 variable x related to each other by the conditions $U_0 \geq w \geq L_0$ when $x = 0$, and $U_1 \geq w \geq L_1$ when $x = 1$. The usual way to model this situation is to use the pair of inequalities

$$U_0 + (U_1 - U_0)x \geq w \geq L_0 + (L_1 - L_0)x. \tag{34}$$

However, when any of the U's or L's are variables, as opposed to constants, cross-product nonlinearities are introduced that must be resolved in order to achieve a suitable linear relaxation. For use in such a case, Peterson (1971) showed that cross-products of the form xz, with z a nonnegative variable bounded above by a constant M and x binary, can be handled by replacing xz with a new variable y which is required to satisfy the conditions

$$Mx \geq y \geq z - M(1-x),\ z \geq y \text{ and } y \geq 0. \tag{35}$$

Assuming that $U_1 \geq U_0$ and $L_1 \geq L_0$, identifying an appropriate upper bound M allows each of the cross products of (34) to be accommodated by introducing a single new variable and three new constraints given by (35), in addition to nonnegativity restrictions.

Glover (1975) specified a more economical approach than Peterson's, while dropping the assumption that $U_1 \geq U_0$ and $L_1 \geq L_0$. This method introduces a total of four new constraints and no new variables. By identifying upper bounds $\overline{U}_0, \overline{L}_0, \overline{U}_1, \overline{L}_1$, and lower bounds $\underline{U}_0, \underline{L}_0, \underline{U}_1, \underline{L}_1$, such that

$$\overline{U}_0 \geq U_0 \geq \underline{U}_0,\quad \overline{L}_0 \geq L_0 \geq \underline{L}_0,\quad \overline{U}_1 \geq U_1 \geq \underline{U}_1,\quad \text{and}\quad \overline{L}_1 \geq L_1 \geq \underline{L}_1, \tag{36}$$

the nonlinear cross-product constraints of (34) can be replaced by the constraints

$$
\begin{aligned}
&U_0 + (\overline{U}_1 - \underline{U}_0)x \geq w \geq L_0 + (\underline{L}_1 - \overline{L}_0)x, \\
&U_1 + (\overline{U}_0 - \underline{U}_1)(1-x) \geq w \geq L_1 + (\underline{L}_0 - \overline{L}_1)(1-x),
\end{aligned} \tag{37}
$$

where $w \in R^1$. When $x = 0$ or $x = 1$, (34) are enforced by one pair of inequalities of (37), with the remaining constraints becoming redundant.

This new linearization was shown by Glover (1975) to easily accommodate quadratic capital budgeting problems seeking to minimize $\sum_{i,j \in N} x_i d_{ij} x_j$ subject to linear constraints, where x_i binary for all $i \in N = \{1, \ldots, n\}$. Defining $w_i = x_i \sum_j d_{ij} x_j$ reduces the number of new variables to n, as opposed to the previous number $n(n-1)/2$ required by the linearization of Peterson. Letting

$$
\begin{aligned}
&\overline{U}_0 = \underline{U}_0 = \overline{L}_0 = \underline{L}_0, \\
&\overline{U}_1 = \overline{L}_1 = D_i^+ = \text{ the sum over } j \text{ of all positive } d_{ij}, \\
&\underline{U}_1 = \underline{L}_1 = D_i^- = \text{ the sum over } j \text{ of all negative } d_{ij},
\end{aligned} \tag{38}
$$

the constraints of (37) become

$$D_i^+ x_i \geq w_i \geq D_i^- x_i, \tag{39}$$

$$\sum_j d_{ij} x_j - D_i^-(1 - x_i) \geq w_i \geq \sum_j d_{ij} x_j - D_i^+(1 - x_i), \tag{40}$$

for $w = w_i$ and $x = x_i$. This linearization was also shown to easily accommodate the more general case of an objective function of the form $\sum_{i \in M} \sum_{j \in N} x_i d_{ij} y_j$ where, as before, the x_i are 0-1 but the variables y_j need not be so constrained. The procedure for handling this is the same as in (39) and (40), redefining D_i^+ and D_i^- appropriately to provide upper and lower bounds for $\sum_j d_{ij} y_j$.

Kettani and Oral (1990) introduced a refinement of the technique of Glover (1975) for bilinear quadratic integer problems of the form

$$
\begin{aligned}
\text{(BQP)} \qquad & \text{Minimize } \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} x_i x_j \\
& \text{subject to: } \sum_{i=1}^{n} a_{ik} x_i \geq b_k \qquad \text{for } k = 1, \ldots, m, \\
& \qquad\qquad x_i = 0 \text{ or } 1 \qquad \text{for } i = 1, \ldots, n.
\end{aligned}
$$

$x_i = 0$	$x_i = 1$
$v_i = 0$	$v_i = \sum_j d_{ij}x_j - D_i^-$
$u_i = 0$	$u_i = D_i^+ - \sum_j d_{ij}x_j$
$z_i = D_i^+ - \sum_j d_{ij}x_j$	$z_i = 0$
$y_i = \sum_j d_{ij}x_j - D_i^-$	$y_i = 0$

TABLE 4. Slack relationships.

This is identical in formulation to the quadratic capital budgeting problem used by Glover (1975). Their refinement reduced the number of new linear constraints needing to be added from $4n$ to $2n$, while keeping the number of new variables to n but constrained in sign. The underlying idea of this linearization was that a system of equations would be more useful than a system of inequalities in identifying redundant conditions.

Introducing slack variables u_i, v_i, z_i, and y_i into Glover's equations (39) and (40), resulted in the equality system

$$w_i = D_i^- x_i + v_i \qquad \forall\, i, \tag{41}$$

$$w_i = D_i^+ x_i - u_i \qquad \forall\, i, \tag{42}$$

$$w_i = \sum_j d_{ij}x_j - D_i^+(1 - x_i) + z_i \qquad \forall\, i, \tag{43}$$

$$w_i = \sum_j d_{ij}x_j - D_i^-(1 - x_i) - y_i \qquad \forall\, i, \tag{44}$$

$$v_i \geq 0, \qquad u_i \geq 0, \qquad z_i \geq 0, \qquad y_i \geq 0, \qquad \forall\, i. \tag{45}$$

Note that the values of the slack variables are determined by the values of the w_i's, which are in turn determined by the values of the x_i's. As can be seen from Table 4 however, $v_i = 0$ *determines* the values of the other slack variables via the system (41)–(45). Therefore, this system of equations can be equivalently replaced by the nonlinear constraints

$$v_i = \left(\sum_j d_{ij}x_j - D_i^-\right)x_i \qquad \forall\, i. \tag{46}$$

Doing so, however, would unnecessarily increase the difficulty of the problem. An equivalent linear representation of the conditions in Table 4 is possible, observing that $v_i + u_i = 0$ implies $v_i = 0$

and $u_i = 0$ since both variables are nonnegative. A similar statement holds true for $z_i + y_i = 0$. Using these relationships to surrogate the expressions in Table 4, yields the linear representation

$$\begin{aligned} & v_i + u_i = (D_i^+ - D_i^-)x_i && \forall i, \\ & z_i + y_i = (D_i^+ - D_i^-)(1 - x_i) && \forall i, \\ & v_i \geq 0, \qquad u_i \geq 0, \qquad z_i \geq 0, \qquad y_i \geq 0, && \forall i. \end{aligned}$$

As a result of adding these to the equality system (41)–(45), equations (42) and (44) could be dropped as redundant. Moreover, the system could be changed back to an inequality system by eliminating all the slack variables except v_i. This gives the linear inequality representation of (41) in $2n$ new constraints and n new variables

$$\begin{aligned} \text{(KO)} \qquad & \text{Minimize } \sum_i (D_i^- x_i + v_i) \\ & \text{subject to: } \sum_j d_{ij} x_j + (D_i^+ - D_i^-)x_i - v_i \leq D_i^+ \qquad \forall i, \\ & v_i \leq (D_i^+ - D_i^-)x_i \qquad \forall i, \\ & x_i \in \{0,1\}, \qquad v_i \geq 0, \qquad \forall i. \end{aligned}$$

One disadvantage of this linearization is that it represents only one particular formulation, since the representation (41) was dependent upon the way substitutions were made during the process of linearization. Even more important, a reduction in the number of constraints defining the linearization of (41) may not be advisable. As was demonstrated earlier, it is far more important to identify and generate the right type of constraints (preferably, valid facetial inequalities) to add to the problem to achieve a certain level of tightening, with the ideal situation being an entire convex hull description. For binary quadratic programming problems, reducing the number of constraints defining the problem without first preprocessing the set in the spirit of Crowder, et al. (1983), introduces the risk of discarding constraints that might have produced effective lifted minimal cover or $(1,k)$-configuation constraints, among others.

Torres (1991) demonstrated that when a mixed cross-product is in the objective function of a minimization problem, or in an inequality of the form $g(x,y) + x_i \cdot F(y) \leq 0$, where $x_i \in \{0,1\}$, $y \in R^n$, the number of constraints necessary for the linearization of Glover (1975) can be further

reduced. For the conditions noted, the constraints given by

$$w \leq F(y) - L \cdot (1 - x_i),$$
$$w \leq U \cdot x_i, \tag{47}$$

where $L \leq F(y) \leq U$ for all feasible $y \in R^n$, are unnecessary. Similarly, if the mixed cross-product is in an inequality constraint of the opposite sense, then the constraints of Glover corresponding to

$$w \leq F(y) - U \cdot (1 - x_i)$$
$$w \leq L \cdot x_i \tag{48}$$

are unnecessary.

Oral and Kettani (1992) introduced a linearization strategy for the same bilinear quadratic programming problem that is a direct algebraic consequence of the linearization of Glover (1975). However, this fact was not readily apparent because they chose to make the cost matrix $D = [d_{ij}]$ symmetric, altering the notation slightly. By retaining similar notation, the connection to Glover's earlier work is clear.

Instead of incorporating the usual linearization substitution $w_i = x_i \sum_j d_{ij} x_j$ of Glover (1975), they noticed that by setting w_i equal to the lower bound specified in (40), i.e.,

$$w_i = \sum_j d_{ij} x_j - D_i^+ (1 - x_i), \tag{49}$$

the required linearization conditions

$$w_i = \begin{cases} 0 & \text{if } x_i = 0, \\ \sum_j d_{ij} x_j & \text{if } x_i = 1, \end{cases} \tag{50}$$

were satisfied when $x_i = 1$, without having to use Glover's other $2n$ constraints given by (39). However, when $x_i = 0$, (49) yielded

$$w_i = \sum_j d_{ij} x_j - D_i^+ (1 - x_i) = \sum_j d_{ij} - D_i^+ \leq 0,$$

and not necessarily equal to zero, a condition satisfied automatically by including (39) in the constraint set. To compensate, they added a quantity $\zeta_i \geq 0$ to (49), giving

$$w_i = \sum_j d_{ij} x_j - D_i^+ (1 - x_i) + \zeta_i, \tag{51}$$

along with the set of constraints

$$\zeta_i \geq -\sum_j d_{ij}x_j + D_i^- x_i + D_i^+(1-x_i) \qquad \forall i,$$

$$\zeta_i \geq 0 \qquad \forall i,$$

that assured the conditions of (50) were satisfied. Thus, the bilinear programming problem could equivalently be formulated as:

(BLP-OK) Minimize $\sum_i \left(\sum_j d_{ij}x_j - D_i^+(1-x_i) + \zeta_i \right)$

subject to: $\zeta_i \geq -\sum_j d_{ij}x_j + D_i^- x_i + D_i^+(1-x_i) \qquad \forall i,$

$$\sum_i a_{ik}x_i \geq b_k \qquad \forall k,$$

$$x_i \in \{0,1\}, \qquad \zeta_i \geq 0 \qquad \forall i.$$

The minimization drives ζ_i toward zero, insuring that for each i, one of the two constraints involving ζ_i will be binding, and hence, that the linearization conditions of (50) are met:

$$\text{if} \quad x_i = 0, \quad \text{then } \zeta_i = -\sum_j d_{ij}x_j + D_i^+,$$

$$\text{if} \quad x_i = 1, \quad \text{then } \zeta_i = 0.$$

While these linearization strategies are concerned with (a) nonlinear 0-1 programs, and (b) preventing a "blow up" in problem size due to the additional variables or constraints being added to a problem formulation, a tightening of the formulation within any hierarchical structure or the treatment of linear 0-1 situations is noticeably absent from consideration. Achieving such a hierarchical tightening requires an additional concept of first reformulating a linear 0-1 program into an equivalent polynomial programming problem, and then linearizing this resulting problem in the spirit of the strategies mentioned.

We should note that Glover and Woolsey were the first to introduce the following, now well known, representation of the cross-product x_1x_2, $x_1, x_2 \in \{0,1\}$, by the continuous variable x_{12}

via the constraints

$$\begin{aligned} x_1 + x_2 - x_{12} &\leq 1, \\ x_1 &\geq x_{12}, \\ x_2 &\geq x_{12}, \\ 0 \leq x_{12} &\leq 1. \end{aligned} \tag{52}$$

Sherali and Adams (1986) observed that the first three constraints of (52) can equivalently be obtained using the bound factor products $(1-x_1)\cdot(1-x_2) \geq 0$, $(1-x_2)\cdot x_1 \geq 0$, and $(1-x_1)\cdot x_2 \geq 0$, respectively, a necessary observation for RLT. Whereas these earlier linearization constructions obtained their linear programming relaxations via a construction of linear constraints specific to each problem formulation, Sherali and Adams (1989) successfully generalized this procedure within a hierarchy of relaxations, taking advantage of the multiplicative nature of constraint generation via bound factor products.

2.2. Hierarchy of Sherali and Adams

To motivate the new hierarchy of relaxations that exploits existing special structures, it is instructive to present the hierarchy of relaxations introduced by Sherali and Adams (1989), along with several other strategies that have been developed subsequently. For the purpose of discussion, we consider the feasible region for the mixed-integer problem defined as

$$X = \{(x,y) \in R^n \times R^m : \; Ax + Dy \geq b, \; 0 \leq x \leq e_n, \; x \text{ binary }, \; y \geq 0\}, \tag{53}$$

where e_n is a vector of ones in R^n. Note that, for convenience, we have assumed that any upper bounds on y-variables, if they exist, are incorporated within the constraints $Ax + Dy \geq b$.

To construct the relaxation at any level $d \in \{1, \ldots, n\}$, define the *bound-factor products* of order d in the following manner:

$$F_d(J_1, J_2) = \left[\prod_{j \in J_1} x_j\right]\left[\prod_{j \in J_2} (1 - x_j)\right] \quad \forall \, J_1, J_2 \subseteq N, \; J_1 \cap J_2 = \emptyset, \; |J_1 \cup J_2| = d.$$

The d^{th} level relaxation X_d can then be explicitly written algebraically using the following reformulation-linearization technique, abbreviated **RLT0** for reference.

Sherali-Adams Reformulation-Linearization Technique (RLT)

(a) **Reformulation.** Multiply each inequality defining the feasible region (53) by each bound–factor product $F_d(J_1, J_2)$ of order d, and apply the identity $x_j^2 = x_j$ for all $j \in \{1, \ldots, n\}$.

(b) **Linearization.** Linearize the resulting polynomial program by substituting

$$w_J = \prod_{j \in J} x_j \; \forall \, J \subseteq N, \quad v_{Jk} = y_k \prod_{j \in J} x_j \; \forall \, J \subseteq N, \; \forall \, k. \tag{54}$$

Then, denoting the projection of X_d onto the space of the original variables (x, y) by

$$X_{P_d} = \{(x, y) : \; (x, y, w, v) \in X_d\} \qquad \forall \, d = 1, \ldots, n, \tag{55}$$

the hierarchy of relaxations

$$X_{P_0} \equiv X_0 \supseteq X_{P_1} \supseteq X_{P_2} \supseteq \cdots \supseteq X_{P_n} = \text{ conv } (X), \tag{56}$$

is obtained, where $X_{P_0} \equiv X_0$ (for $d = 0$) denotes the ordinary linear programming relaxation, and conv (X) is the convex hull of X. Note that here, $w \equiv (w_J, \; J \subseteq N, \; |J| \geq 2)$ and $v \equiv (v_{Jk}, \; J \subseteq N, \; |J| \geq 1, \; k \in \{1, \ldots, m\})$. In particular, note also that $w_\emptyset \equiv 1$, $w_J \equiv x_j$ for $J = \{j\}$, $j = 1, \ldots, n$, and $v_{Jk} \equiv y_k \; \forall \, k = 1, \ldots, m$ when $J = \emptyset$. Hence, we can also assume that $w \equiv (w_J, \; J \subseteq N)$ and $v \equiv (v_{Jk}, \; J \subseteq N, \; k = 1, \ldots, m)$.

For convenience in their analysis, Sherali and Adams assume that (scaled) upper bounds $y \leq e_m$ exist on the y–variables. However, this assumption is not necessary, and the proof of their main Theorem 2 holds simply by eliminating the corresponding product terms and constraints. In particular, given feasibility of the underlying mixed–integer program, their proof asserts that x is binary valued at an optimum to any linear program $\max\{cx + dy : \; (x, y) \in X_{P_n}\}$ for which there exists an optimal solution, i. e. , for which there does not exist any $\delta \neq 0$ such that $D\delta \geq 0$, $\delta \geq 0$, and $d^T \delta > 0$.

One can think of the hierarchy defined by (56) as being constructed inductively, where, given the relaxation at level $d \in \{0, \ldots, n\}$, the relaxation at level $(d+1)$ is constructed by multiplying each constraint of X_d by each x_t and $(1 - x_t)$ for $t = 1, \ldots, n$, and then re–linearizing. Note that this interpretation is conceptual in nature. In an actual implementation of such an inductive scheme, many products will produce implied or null constraints at level $(d + 1)$. For example, given a constraint that has been generated at level d via the factor $F_d(J_1, J_2)$ being applied to a

constraint defining X, when this is multiplied by x_t to produce a level $(d+1)$ constraint for some $t \in \{1, \ldots, n\}$, we would have the following cases:

(*i*) $t \in J_1$. This reproduces the same constraint as at level d, since $F_d(J_1, J_2) \cdot x_t = F_{(d-1)}(J_1 - \{t\}, J_2) \cdot x_t^2 = F_d(J_1, J_2)$ under the substitution $x_t^2 = x_t$. Moreover, this level d constraint is demonstrated by Sherali and Adams to be implied by the other level $(d+1)$ constraints, and therefore can be suppressed.

(*ii*) $t \in J_2$. This produces a null constraint since $F_d(J_1, J_2) \cdot x_t = F_{(d-1)}(J_1, J_2 - \{t\}) \cdot (1 - x_t) \cdot x_t = 0$ under the substitution $x_t^2 = x_t$.

(*iii*) $t \notin J_1 \cup J_2$. This produces a level $(d+1)$ constraint as would be generated via the Reformulation–Linearization Technique **RLT0** stated above.

A similar argument holds for a product using $(1 - x_t)$ for $t \in \{1, \ldots, n\}$.

2.3. Other reformulation-linearization strategies

Based on disjunctive programming methods, Balas (1983) has shown in an implicit, inductive fashion how a hierarchy of relaxations spanning from the linear programming to the convex hull representation could be constructed for mixed–integer 0-1 problems. Disjunctive programming is optimization over disjunctive sets. A *disjunctive set* is a set defined by inequalities connected to each other by the operations of conjunction ($\wedge$, juxtaposition, "and") or disjunction ($\vee$, "or"). Since inequalities define halfspaces, a disjunctive set can also be viewed as a collection of halfspaces joined together by the operations of intersection ($\cap$) or union ($\cup$). A disjunctive program is then a problem of the form $\min\{cx : x \in F\}$, where F is a disjunctive set. Any integer or mixed-integer program can be stated as a disjunctive program, usually in more than one way. Conversely, any bounded disjunctive program can be stated as a mixed-integer program.

Balas' basic disjunctive programming construct is a description of the closure convex hull of a union of polyhedral sets in terms of a certain higher dimensional polyhedron. Given a representation of the feasible region as a set in conjunctive normal form $F = \bigcap_{j \in T} S_j$, where each S_j is a union of polyhedra for each j in the index set T, a relaxation of the closure convex hull of F is obtained by taking the *hull-relaxation* of F, defined by the intersection of the closure convex

hull of each S_j, i.e.,

$$\text{h-rel } F \equiv \bigcap_{j \in T} \text{cl conv } S_j. \tag{57}$$

The hull relaxation of F is not to be confused with the convex hull of F; its usefulness comes from the fact that it involves taking the convex hull of each union of polyhedra before intersecting them. If F is written as the intersection of its continuous relaxation constraints, along with the disjunctions that $x_j = 0$ or $x_j = 1$ for each binary variable x_j, this process yields the usual linear programming relaxation. The hierarchy of relaxations results from the process of converting in a stage-wise fashion the intersections of each of the various pairs of sets S_j to an equivalent union of polyhedra before taking the hull-relaxation. For example, for $j = 1, 2$, let $S_j = \bigcup_{i \in Q_j} P_i$, where each P_i, $i \in Q_j$, $j = 1, 2$, is a polyhedron. Then,

$$\text{cl conv } (S_1 \cap S_2) \subseteq (\text{cl conv } S_1) \cap (\text{cl conv } S_2). \tag{58}$$

On the other end of the spectrum, if F is represented as a union of polyhedra ($|T| = 1$) by essentially enumerating all possible feasible binary combinations, the closure convex hull of F is recovered. For $i = 0, 1, \ldots, t$, letting $F_i = \bigcap_{j \in T_i} S_j$ be a sequence of conjunctive normal representations of a disjunctive set, then

$$P_0 = \text{h-rel } F_0 \supseteq \text{h-rel } F_1 \supseteq \cdots \supseteq \text{h-rel } F_t = \text{cl conv } F, \tag{59}$$

where P_0 denotes the usual linear programming relaxation.

A related hierarchy of relaxations was independently developed by Lovasz and Schrijver (1991) which amounts to deriving a first level tightening X_1 over the linear programming relaxation X_0, finding the projection X_{P1}, and then repeating this step by replacing X_0 with X_1. Continuing in this fashion, they demonstrated that in n steps, the convex hull of X is obtained. As mentioned in the previous chapter, their requirement of interspersing an explicit projection operation, however, renders their procedure as impractical for computational purposes.

Balas, Ceria, and Cornuéjols (1993) later studied a special case of the Sherali–Adams (1989) RLT procedure by imposing integrality on the variables one at a time, *i.e.*, by sequentially constructing Sherali and Adams' n^{th} level relaxation one variable at a time.

Balas, Ceria, and Cornuéjols' Lift-and-Project Procedure

(a) **Reformulation.** Select an index $j \in \{1,\ldots,p\}$. Multiply each inequality of the constraint set $Ax \geq b$ by x_j and $(1-x_j)$, to obtain the nonlinear system

$$(1-x_j)(Ax-b) \geq 0,$$
$$x_j(Ax-b) \geq 0,$$
$$x_j(1-x_i) \geq 0,$$
$$x_jx_i \geq 0.$$

(b) **Linearization.** Linearize the resulting polynomial program by substituting y_i for x_ix_j, $i = 1,\ldots,n$, $i \neq j$, and x_j for x_j^2, $j = 1,\ldots,p$. Denoting the polyhedron defined by this system $M_j(K)$, project $M_j(K)$ into the x-space by eliminating y_i, $i = 1,\ldots,n$, $i \neq j$. Call this resulting polyhedron $P_j(K)$.

Notice that this lift-and-project construction is primarily existential, since the computational cost of imposing a projection operation at each stage of iteration is prohibitive. Note also that the intermediate level relaxations of Sherali and Adams (1989) are different in structure, and moreover, are algebraically explicit. This facilitates their use in linear programming based branch–and–bound/cut algorithms as well as provides a construct for deriving classes of facets. Additionally, both the Sherali-Adams relaxation X_{P_d}, and the Lovasz-Schrijver relaxation (denoted by $N(K)$ in their paper), are not only stronger than $P_j(K)$ for any j, but also stronger than $\bigcap_{j=1}^{p} P_j(K)$; the inclusions $X_{P_d} \subseteq \bigcap_{j=1}^{p} P_j(K)$ and $N(K) \subseteq \bigcap_{j=1}^{p} P_j(K)$ can be strict.

A direct consequence of the ability of RLT to recover the convex hull of X is that the convex hull representation over subsets of the x-variables can be computed in an identical fashion. In particular, suppose that the first $p \leq n$ variables are treated as binary, with the remaining $(n-p)$ x-variables relaxed to be continuous. It directly follows that the p^{th} level linearization of this relaxed problem produces $\text{conv}\{X_0 \cap \{(x,y) : x_j \text{ is binary } \forall\, j = 1,\ldots,p\}\}$. Repeating this procedure for the remaining $(n-p)$ x-variables appropriately recovers $\text{conv}(X)$. The selection of variables might be guided by their relative importance within the framework of the problem formulation. An alternative would be to apply such a scheme on a subset of binary variables that turn out to be fractional in the initial linear programming relaxation.

This observation suggests employing an RLT procedure guided by some other explicit special

structures present in the given problem formulation, such as set packing constraints, or some implicit constraints that provide polyhedral information, such as minimal cover constraints. Using such a technique, it would be possible to generate tight valid inequalities implied by higher level relaxations not evident in the original formulation, or explicitly generate convex hull representations or facetial inequalities by applying the highest level RLT scheme to various subsets of constraints that involve a manageable number of variables. Sherali and Lee (1993) effectively employed such an approach for the set partitioning problem, as explained in the next section.

2.4. Partial convex hull representations

Constructing a partial first-level relaxation alone has shown to yield encouraging computational results. For the maximum clique problem, Balas et al. (1994) have recently observed that by imposing integrality only on the variables involved in a single clique constraint involving n variables, an application of the corresponding n^{th} level relaxation of Sherali and Adams (1989) reduces to that achieved at the first-level using multiplicative factors generated from this constraint. This observation reinforces the rationale for explicitly defining a hierarchy of relaxations that would accommodate varying special structures.

This modified lift-and-project algorithm is based on a graph theoretical interpretation of RLT, and exploits the structure of a *stable set*, the graph theoretical equivalent of a GUB constraint. A vertex set is called *stable* if every two vertices in S are nonadjacent. For any stable set S of size greater than two, the constraint $\sum_{i \in S} x_i \leq 1$ is valid for the clique polytope $C(G) \equiv \text{conv}\{x \in \{0,1\}^n : x_i + x_j \leq 1, \forall (i,j) \notin E\}$, where E is the edge set of the graph $G = (V, E)$. Let K_0 be a relaxation of $C(G)$ such that $C(G) = \text{conv}(K)$, where $K = \{x \in \{0,1\}^n : x \in K_0\}$.

Balas et al. (1994) modified their lift-and-project algorithm in the following way to explicitly incorporate multiplicative factors based on the presence of stable sets.

Modified lift-and-project algorithm

(a) **Reformulation.** Multiply each inequality of the constraint set $Ax \geq b$ by $(1 - x(S))$ and

x_i, $i \in S$, to obtain the nonlinear system

$$(1 - x(S))(Ax - b) \geq 0.$$

$$x_i(Ax - b) \geq 0, \qquad i \in S.$$

(b) **Linearization.** Linearize the resulting polynomial program by substituting y_{ij} for $x_i x_j$, $j = 1, \ldots, n$, $i \in S$, and x_i for x_i^2, $i \in S$, and setting $y_{ij} = y_{ji}$. Denoting the polyhedron defined by this system $H_S(K_0)$, project $H_S(K_0)$ into the x-space. Call this resulting polyhedron $R_S(K_0)$.

The algorithm amounts to convexifying over all the variables in S using only a first level linear programming relaxation. Hence, a partial convex hull representation based on imposing integrality on only this partial set of variables is produced at the first level, so that, as shown by Balas et al., $R_S(K_0) = \text{conv}\{K_0 \cap \{x : x_i \in \{0,1\}, \text{ for all } i \in S\}\}$. The proof of this result comes from a Sherali and Adams (1989) n^{th} level RLT application being simplified for this special case. Moreover, this modified lift-and-project algorithm can also be interpreted as a special case of a more general observation that was made by Sherali and Lee (1993) for the set partitioning problem in the following way.

Let $G = (V, E)$ be the intersection graph associated with the set partitioning problem $SP \equiv \min\{cx : Ax = e, x_j \in \{0,1\} \quad \forall j \in N\}$; that is, the vertex set $V = N$ and vertices i and j are connected by an edge (i, j) if and only if $a_i a_j \neq 0$. Assuming that G is both connected and not complete (if G is complete, then problem SP is trivial), a subset W of the vertex set is called an *independent set* of G if no two vertices of W are adjacent in G. An independent set is *maximal* if G has no independent set W' having $|W'| > |W|$. The number of vertices in a maximal independent set of G is called the *independence number* of G, denoted by $\alpha(G)$, and note that $1 < \alpha(G) < n$. Sherali and Lee (1993) showed that since $\sum_{j \in N} x_j \leq \alpha(G)$ for all x feasible to SP, then

$$\prod_{j \in J} x_j = 0 \qquad \forall J \subseteq N \text{ such that } |J| > \alpha(G). \tag{60}$$

That is, all RLT multilinear product terms equal zero whose cardinality exceeds the independence number of the intersection graph, so that the convex hull of SP is recovered by the projection of the $|\alpha(G)|^{th}$ level RLT relaxation, i.e., $SPP = SPP_{P\alpha(G)}$. For the special case demonstrated by

Balas, et al. (1994), the independence number equals one, and hence the result follows.

2.5. Insights into RLT

At each level of RLT, the algebraic relationship that accounts for the tightening of linear programming relaxations is the replacement of terms x_j^2 by x_j. While not eliminating any points for which $x_j \in \{0,1\}$, it does cut off fractional points in which $0 < x_j < 1$. However, it is instructive to understand exactly what RLT is accomplishing with respect to the convex hull of binary integer feasible solutions in order to understand the motivation to seek out and exploit special structures in problem formulations.

For any particular RLT factor used in a specific product, say $x_j \in \{0,1\}$, RLT is implicitly resolving a node 0 branching decisions for x_j on the order of 2^d, where d is the order of the multiplicative factors F_d. To illustrate this particular recursive structure, consider the simple MIP given by

$$\begin{aligned} &\text{Minimize } \sum_{j \in J} c_j x_j + cy \\ &\text{subject to: } \sum_{j \in J} a_j x_j + Ay \geq b, \\ &\qquad y \geq 0, \\ &\qquad x_j \in \{0,1\} \quad \forall j \in J. \end{aligned} \tag{61}$$

RLT selects some subset of J to form multiplicative factors comprised of binary variables x_j and their corresponding complements $(1 - x_j)$ according to the rule that for all $j \in F_d(J)$, x_j and $(1 - x_j)$ cannot appear in the same factor, for else, this factor is simply zero.

RLT multiplies each of the constraints defining (61) by the factors so formed. These factors are sometimes referred to as "words," with each x_j or $(1 - x_j)$ term called an "alphabet" because of the anology of the constructive nature of the factors to varying length words. This multiplication results in a polynomial programming problem that is subsequently linearized by applying the substitution specified in the previous section.

Suppose that $|J| = 1$ so that the two alphabets x_1 and $(1 - x_1)$ are used as RLT multipliers.

We have,

$$\begin{aligned} &\Big(a_1x_1 + Ay \geq b\Big) \cdot x_1, \\ &\Big(a_1x_1 + Ay \geq b\Big) \cdot (1 - x_1), \end{aligned} \tag{62}$$

which yields the polynomial representation (when $x_1 = x_1^2$):

$$\begin{aligned} a_1x_1 + Ayx_1 &\geq bx_1, \\ Ay - Ayx_1 &\geq b(1 - x_1). \end{aligned} \tag{63}$$

Setting $w = yx_1$, we linearize the polynomial problem, yielding

$$\begin{aligned} a_1x_1 + Aw &\geq bx_1, \\ Ay - Aw &\geq b(1 - x_1). \end{aligned} \tag{64}$$

Substituting $z = y - w$, results in the set of constraints

$$\begin{aligned} Aw &\geq (b - a_1)x_1, \\ Az &\geq b(1 - x_1). \end{aligned} \tag{65}$$

For this case of $|J| = 1$, a first level RLT application has transformed the original formulation of MIP given by (61) into a separable linear programming relaxation given by

$$\begin{aligned} \min_{0 \leq x_1 \leq 1} c_1x_1 + \min_{z,w}\{cz + cw\} \\ \text{st:} \quad Aw &\geq (b - a_1)x_1, \\ Az &\geq b(1 - x_1), \\ w &\geq 0, \\ z &\geq 0, \end{aligned} \tag{66}$$

which is decomposible into the form

$$\begin{array}{lllll} \min\limits_{0 \leq x_1 \leq 1} c_1x_1 & + & \min\limits_{z} cz & + & \min\limits_{w} cw \\ & \text{st:} & Az \geq b(1 - x_1) & \text{st:} & Aw \geq (b - a_1)x_1 \\ & & z \geq 0 & & w \geq 0 \end{array} \tag{67}$$

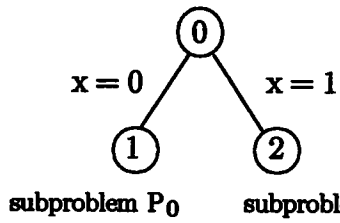

FIGURE 6. (a) Two node branching for RLT.

We can project the problem into the original x-space by factoring x_1 and $(1-x_1)$ from the two minimization subproblems. Thus, we have

$$\min_{0\le x_1\le 1} c_1x_1 + (1-x_1)\underbrace{\left[\begin{array}{ll}\min\limits_{y} cy & \\ \text{st:} & Ay \ge b \\ & y \ge 0\end{array}\right]}_{P_0} + (x_1)\underbrace{\left[\begin{array}{ll}\min\limits_{y} cy & \\ \text{st:} & Ay \ge (b-a_1) \\ & y \ge 0\end{array}\right]}_{P_1} \tag{68}$$

Denoting the value of subproblems P_0 and P_1 as $v(P_0)$ and $v(P_1)$ respectively, we can write (68) as

$$\min_{0\le x_1\le 1}\{c_1x_1 + (1-x_1)v(P_0) + (x_1)v(P_1)\},$$

or,

$$v(P_0) + \min_{0\le x_1\le 1}\{(c_1 - v(P_0) + v(P_1))x_1\}. \tag{69}$$

The representation of (68) and (69) reveals the essence of the RLT procedure. The subproblem P_0 is the projected problem of node 1 in the binary branching shown in Figure 6 that becomes 'active' when $x_1 = 0$. Likewise, the subproblem P_1 is the projected problem of node 2 for the same branching with $x_1 = 1$. Assuming that P_0 and P_1 are both feasible, notice that their feasible regions differ in tightness. For example, when $a_1 \ge 0$, subproblem P_0 is tighter, and vice-versa for $a_1 \le 0$. Hence, for any feasible solution to both P_0 and P_1, the minimization will attempt to activate the more relaxed subproblem, driven by the coefficient $(c_1 - v(P_0) + v(P_1))$ of x_1 in (69). When $(c_1 - v(P_0) + v(P_1)) \ge 0$, the minimization drives $x_1 = 0$, and when $(c_1 - v(P_0) + v(P_1)) \le 0$, it likewise drives $x_1 = 1$.

To further illustrate this idea, if $J = \{1,2\}$, so that problem (61) has two binary variables x_1 and x_2, the level $d = 2$ RLT relaxation problem, which uses RLT multipliers $F_2(\{1,2\},\emptyset) = x_1x_2 \geq 0$, $F_2(\{1\},\{2\}) = x_1(1-x_2) \geq 0$, $F_2(\{2\},\{1\}) = x_2(1-x_1) \geq 0$, and $F_2(\emptyset,\{1,2\}) = (1-x_1)(1-x_2) \geq 0$, yields the set of nonlinear constraints

$$
\begin{aligned}
a_1x_1x_2 + a_2x_1x_2 + Ayx_1x_2 &\geq bx_1x_2,\\
a_1x_1 - a_1x_1x_2 + Ayx_1 - Ayx_1x_2 &\geq bx_1(1-x_2),\\
a_2x_2 - a_2x_1x_2 + Ayx_2 - Ayx_1x_2 &\geq bx_2(1-x_1),\\
Ay - Ayx_1 - Ayx_2 + Ayx_1x_2 &\geq b(1-x_1)(1-x_2),\\
x_1x_2 &\geq 0,\\
x_1 - x_1x_2 &\geq 0,\\
x_2 - x_1x_2 &\geq 0,\\
1 - x_1 - x_2 + x_1x_2 &\geq 0,
\end{aligned}
\tag{70}
$$

where $x_ix_j = x_jx_i$ and $x_i^2 = x_i$, as usual. Letting $w_{12} = x_1x_2$ yields the system

$$
\begin{aligned}
(a_1 + a_2)w_{12} + Ayw_{12} &\geq bw_{12},\\
a_1(x_1 - w_{12}) + Ay(x_1 - w_{12}) &\geq b(x_1 - w_{12}),\\
a_2(x_2 - w_{12}) + Ay(x_2 - w_{12}) &\geq b(x_2 - w_{12}),\\
Ay(1 - x_1 - x_2 + w_{12}) &\geq b(1 - x_1 - x_2 + w_{12}),\\
w_{12} &\geq 0,\\
x_1 - w_{12} &\geq 0,\\
x_2 - w_{12} &\geq 0,\\
1 - x_1 - x_2 + w_{12} &\geq 0.
\end{aligned}
\tag{71}
$$

Now, define $v_J = y\prod_{j\in J} x_j$ in the usual RLT manner, so that $v_1 = yx_1$, $v_2 = yx_2$, and $v_{12} =$

$yx_1x_2 = yw_{12}$. This yields an equivalent linear system

$$
\begin{aligned}
(a_1 + a_2)w_{12} + Av_{12} &\geq bw_{12}, \\
a_1(x_1 - w_{12}) + A(v_1 - v_{12}) &\geq b(x_1 - w_{12}), \\
a_2(x_2 - w_{12}) + A(v_2 - v_{12}) &\geq b(x_2 - w_{12}), \\
A(y - v_1 - v_2 + v_{12}) &\geq b(1 - x_1 - x_2 + w_{12}), \\
1 - x_1 - x_2 + w_{12} &\geq 0, \\
w_{12}, v_1, v_2, y &\geq 0.
\end{aligned} \tag{72}
$$

In practice, there is no need to pursue further linearization of the constraint set, since (72) defines an appropriate higher dimensional polyhedral set X_2 defined in terms of the original variables (x, y) and the new auxiliary variables (w, v).

However, for the purpose of displaying the decomposition of the problem, we can substitute $z_{12} = w_{12}$, $z_1 = x_1 - w_{12}$, $z_2 = x_2 - w_{12}$, $z_0 = 1 - x_1 - x_2 + w_{12}$, and similarly, we can substitute $s_{12} = v_{12}$, $s_1 = v_1 - v_{12}$, $s_2 = v_2 - v_{12}$, and $s_0 = y - v_1 - v_2 + v_{12}$. The inverse transformation yields $w_{12} = z_{12}$, $x_1 = z_1 + z_{12}$, $x_2 = z_2 + z_{12}$, where $z_0 + z_1 + z_2 + z_{12} = 1$, and similarly, $v_{12} = s_{12}$, $v_1 = s_1 + s_{12}$, $v_2 = s_2 + s_{12}$, and $y = s_0 + s_1 + s_2 + s_{12}$. Substituting these into (72), along with the objective function in (61), yield the equivalent problem (in continuous variables)

$$
\begin{aligned}
\text{Minimize} \quad & c_1(z_1 + z_{12}) + c_2(z_2 + z_{12}) + c(s_0 + s_1 + s_2 + s_{12}) \\
\text{subject to:} \quad & (a_1 + a_2)z_{12} + As_{12} \geq bz_{12}, \\
& a_1 z_1 + As_1 \geq bz_1, \\
& a_2 z_2 + As_2 \geq bz_2, \\
& As_0 \geq bz_0, \\
& z_0 + z_1 + z_2 + z_{12} = 1, \\
& (z, s) \geq 0.
\end{aligned} \tag{73}
$$

This linear program decomposes as follows.

$$\min_{\substack{z:\ z_0+z_1+z_2+z_{12}=1 \\ z\geq 0}} \quad c_1z_1 + c_2z_2 + \min_{s_0} cs_0 \quad + \quad \min_{s_1} cs_1$$

$$\text{st: } As_0 \geq bz_0 \qquad \text{st: } As_1 \geq (b-a_1)z_1$$

$$s_0 \geq 0 \qquad s_1 \geq 0$$

(74)

$$+ \quad \min_{s_2} cs_2 \qquad + \quad \min_{s_{12}} cs_{12}$$

$$\text{st: } As_2 \geq (b-a_2)z_2 \qquad \text{st: } As_{12} \geq (b-a_1-a_2)z_{12}$$

$$s_2 \geq 0 \qquad s_{12} \geq 0$$

$$(75) \qquad \equiv \min_{\substack{z:\ z_0+z_1+z_2+z_{12}=1 \\ z\geq 0}} \quad z_0 \left[\begin{array}{c} \min_y cy \\ \text{st: } Ay \geq b \\ y \geq 0 \end{array}\right] + z_1 \left[\begin{array}{c} c_1 + \min_y cy \\ \text{st: } Ay \geq (b-a_1) \\ y \geq 0 \end{array}\right]$$

$$+ \quad z_2 \left[\begin{array}{c} c_2 + \min_y cy \\ \text{st: } Ay \geq (b-a_2)y \\ y \geq 0 \end{array}\right] + z_{12} \left[\begin{array}{c} c_1 + c_2 + \min_y cy \\ \text{st: } Ay \geq (b-a_1-a_2) \\ y \geq 0 \end{array}\right]$$

Note that this is equivalent to solving the following decomposed problem, which simply enumerates all possible combinations.

$$(76) \quad \min_{x_1,x_2 \text{ binary}} \quad (1-x_1)(1-x_2) \underbrace{\left[\begin{array}{c} \min_y cy \\ \text{st: } Ay \geq b \\ y \geq 0 \end{array}\right]}_{P_{00}} + x_1(1-x_2) \underbrace{\left[\begin{array}{c} c_1 + \min_y cy \\ \text{st: } Ay \geq (b-a_1) \\ y \geq 0 \end{array}\right]}_{P_{10}}$$

$$+ \quad x_2(1-x_1) \underbrace{\left[\begin{array}{c} c_2 + \min_y cy \\ \text{st: } Ay \geq (b-a_2)y \\ y \geq 0 \end{array}\right]}_{P_{01}} + x_1x_2 \underbrace{\left[\begin{array}{c} c_1 + c_2 + \min_y cy \\ \text{st: } Ay \geq (b-a_1-a_2) \\ y \geq 0 \end{array}\right]}_{P_{11}}.$$

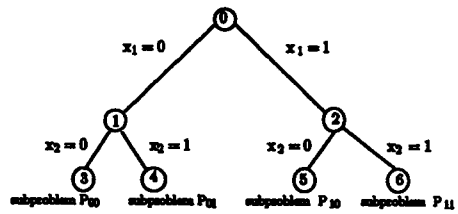

FIGURE 7. (a) Level $d = 2$ node branching for RLT.

Hence, the linear program whose objective is to minimize $(c_1x_1 + c_2x_2 + cy)$ subject to the region defined by (72) would automatically solve (61).

The RLT procedure for level $d = 2$ represents an implicit decomposition of the minimization problem (61) that can be characterized explicitly in terms of evaluating the contribution of each subproblem at node 0 to resolving the decisions at nodes $1 - 6$ illustrated in Figure 7. RLT relaxations for levels $d > 1$ extend the depth and number of binary subproblems being resolved at node 0. By selecting the level of RLT relaxations to be $d = n$, all binary branching decisions for the integer portion of the MIP are being resolved at node 0 along with the continuous variable values, thus enabling the linear programming relaxation for P_{X_n} to recover the convex hull of binary integer feasible solutions exactly.

For the case where only a single binary variable exists for (61), notice that a level $d = 1$ relaxation resolves $x_1 = 0$ or $x_1 = 1$, and hence produces the partial convex hull conv$\{(y, x_j, j \in J)$: constraints of (61) with $y \geq 0$, $0 \leq x_j \leq 1$, $j \in J - \{1\}$, x_1 binary $\}$ using only bound factor products $x_1 \geq 0$ and $(1 - x_1) \geq 0$. Using multiplicative factors at the same level that involve more than a single binary variable serves to tighten the relaxation to the intersection of such partial convex hulls. A use of multiplicative constraint factors in addition to the foregoing bound factor products further tightens the relaxations thus produced. In particular, if an MIP possesses some special structure, either explicit or implied, this strategy allows stronger RLT factors to be created that can attack fractionating propensities of more than one binary variable at lower levels of the hierarchy. This is precisely the philosophy that motivates the new hierarchy of relaxations presented in Chapter 3.

CHAPTER 3

New Hierarchy of Relaxations

In this chapter, we present a new hierarchy of relaxations that provides a unifying framework for constructing a spectrum of continuous relaxations spanning from the linear programming relaxation to the convex hull representation for linear mixed integer 0-1 problems. This hierarchy is an extension of the Reformulation-Linearization Technique (RLT) of Sherali and Adams (1989, 1990, 1994), and is particularly designed to exploit special structures. We demonstrate that inherent special structures can be exploited to develop specific classes of multiplicative factors that can be applied to the original problem to reformulate it as an equivalent polynomial programming problem, that is subsequently linearized to produce a tighter relaxation in a higher dimensional space.

We then present several specific examples to demonstrate how underlying special structures, including generalized upper bounding (GUB), variable upper bounding (VUB), set covering, set partitioning, and set packing constraints, as well as sparsity, can be exploited within this framework. We also introduce an alternative partial application strategy for this new hierarchy, designed to reduce the growth in number of constraints necessary to be added to the reformulation of a problem. A new class of *conditional logical* constraints directly obtainable within the framework of this new hierarchy is then presented, along with establishing the equivalence of these conditional logical constraints with those obtainable through sequential lifting methods. Finally, we identify a class of constraints that can effectively be deleted from certain problem formulations, and demonstrate how implied constraints in the projection of such a **RLT1**-based higher dimensional problem are not implied in the higher dimension itself.

For convenience in notation in presenting the development of this new hierarchy, let us denote by $\{\cdot\}_L$ the process of linearizing a polynomial expression $\{\cdot\}$ in x and y via the substitution defined in (54), following the use of the identity $x_j^2 = x_j \; \forall \; j = 1, \ldots, n$. Accordingly, let us make the following observation that can be readily verified algebraically. Consider any pair of polynomial expressions Ψ and Φ. Then, the following operation is valid:

$$\{\Psi\}_L + \{\Phi\}_L = \{\Psi + \Phi\}_L. \tag{77}$$

Moreover, whenever we multiply a linearized expression with a polynomial factor, we will recognize the corresponding polynomial terms that were linearized in the former expression, and hence treat this product as being equivalent to multiplying the corresponding polynomial expression that was linearized with this latter factor. More succinctly, we will assume that

$$[\{\Psi\}_L \cdot \{\Phi\}]_L \equiv [\{\Psi\} \cdot \{\Phi\}]_L, \tag{78}$$

where $(\cdot)$ denotes the usual (algebraic) product.

Consider an alternate equivalent representation of the feasible region (53) written in the form

$$X = \{(x, y) : \; \tilde{A}x + \tilde{D}y \geq \tilde{b}, \; g_i x - g_{0i} \geq 0 \;\text{ for }\; i = 1, \ldots, p \,, \; y \geq 0, \; x \text{ binary } \}, \tag{79}$$

where the set of constraints defining $S \equiv \{x : \; g_i x - g_{0i} \geq 0 \quad \forall \, i = 1, \ldots, p\}$ are either constraints that might have been originally present in the set X represented in (53), or may be valid inequalities implied by X of (53). (Several examples of such sets S will be considered in the sequel, leading to different ideas for exploiting inherent special structure.) More importantly, we assume that for each $t = 1, \ldots, n$,

$$\min\{x_t \; : \; x \in S\} = 0 \quad \text{and} \quad \max\{x_t \; : \; x \in S\} = 1. \tag{80}$$

Note that if $\min(x_t) > 0$, we can fix $x_t = 1$, and if $\max(x_t) < 1$, we can fix $x_t = 0$, and if both these conditions hold for any t, then the problem is infeasible. Therefore, we will assume that the equalities of (80) hold for all t. Hence, for each $t = 1, \ldots, n$, denoting λ_i^t, $i = 1, \ldots, p$, as the set of optimal dual multipliers for the first (min) problem in (80), we have by dual feasibility that $(\sum_{i=1}^p \lambda_i^t g_i)\, x = x_t$, while the optimal dual objective function value $\sum_{i=1}^p \lambda_i^t g_{0i} = 0$. This gives,

$$\sum_{i=1}^{p} \lambda_i^t (g_i x - g_{0i}) = x_t, \quad \text{where } \; (\lambda_i^t, i = 1, \ldots, p) \geq 0, \; \forall \, t = 1, \ldots, n. \tag{81}$$

Similarly, for each $t = 1, \ldots, n$, denoting π_i^t, $i = 1, \ldots, p$, as the set of optimal dual multipliers for the second problem in (80), we get,

$$\sum_{i=1}^{p} \pi_i^t (g_i x - g_{0i}) = (1 - x_t), \quad \text{where } (\pi_i^t, i = 1, \ldots, p) \geq 0, \ \forall\, t = 1, \ldots, n. \tag{82}$$

Now, define the sets P and $\bar{P}$ as follows, where $\bar{P}$ duplicates each index in P n times:

$$P = \{1, \ldots, p\}, \ \text{ and } \ \bar{P} = \{n \text{ copies of } P\}. \tag{83}$$

3.1. Constructing the new hierarchy

The construction of the new hierarchy proceeds in a manner similar to that employed by Sherali and Adams. At any chosen level of relaxation $d \in \{1, \ldots, n\}$, we construct a higher dimensional relaxation $\bar{X}_d$ by considering the *S–factors of order d* defined as follows:

$$g(J) = \prod_{i \in J} (g_i x - g_{0i}) \qquad \text{for each distinct } J \subseteq \bar{P}, \ |J| = d. \tag{84}$$

Using $(d-1)$ suitable dummy indices to represent duplications, it can be shown that there are a total of $\binom{p+d-1}{d} = (p+d-1)!/d!(p-1)!$ distinct factors of this type at level d. These factors are then used in a reformulation–linearization technique, abbreviated **RLT1**, as stated below, in order to generate the relaxation $\bar{X}_d$.

(a) **Reformulation.** Multiply each inequality defining the feasible region (79) by each $g(J)$ of order d, and apply the identity $x_j^2 = x_j$ for all $j \in \{1, \ldots, n\}$.

(b) **Linearization.** Linearize the resulting polynomial program by using the substitution defined in (54). This produces the d^{th}–level relaxation $\bar{X}_d$.

As in (55), define the projection

$$\bar{X}_{P_d} = \{(x, y) \ : \ (x, y, w, v) \in \bar{X}_d\} \quad \forall\, d = 1, \ldots, n. \tag{85}$$

Before proceeding to analyze the fundamental properties of the relaxations produced by **RLT1** in relation to **RLT0**, let us highlight some important comments that pertain to the application of **RLT1** in particular instances via the following remarks.

Remark 1. In an actual implementation, note that under the substitution $x_j^2 = x_j$ for all j, several terms defining the factors in (84) might be zeros. Also, some of these factors might also be

implied in the sense that they can be reproduced as a nonnegative surrogate of other such factors that are generated in (84). All such null and implied factors and terms should be eliminated before applying **RLT1**. As an aside, note that if all such combinations of factors are constructed in (84), the total number of distinct factors of type $g(J)$ that can be generated at level d is given by

$$\sum_{k=0}^{d} \binom{n+k-1}{k} \binom{n+(d-k)-1}{d-k}.$$

■

Remark 2. We could think of **RLT1** as being an inductive process, with the relaxation at level $(d+1)$ being produced by multiplying each of the constraints in the relaxation at level d with each constraint defining S. Constraints produced by this process that effectively use null (zero) factor expressions $g(J)$ of order d are null constraints. Constraints produced by this process that effectively use factors $g(J)$ that are implied by other factors in (84), are by virtue of (78) and (77), themselves implied by the constraints generated using the latter factors which imply such factors. Hence, the process of reducing the set of factors as in Remark 1, or that of eliminating the corresponding redundant constraints generated by such factors, are equivalent steps. For convenience in analysis, we assume that a full relaxation using all possible factors of order d is generated at level d. By the foregoing comments, it follows that an equivalent relaxation at level d would be produced by using only the factors identified in Remark 1, recognizing any zero variable terms in the resulting relaxation as identified by the S–factors. Furthermore, such non–redundant/non–null factors can be generated inductively through the levels, recognizing zero terms revealed at previous levels. This latter relaxation is what should actually be generated in practice. ■

To state our main results, let us now suppose that we apply the usual RLT process of Sherali and Adams (**RLT0**) to the feasible region X as defined in (79). This is accomplished by assuming that the inequalities $0 \leq x \leq e_n$ have been explicitly incorporated into X, and that the usual multiplicative factors $F_d(J_1, J_2)$ for all (J_1, J_2) of order d are applied to all the defining inequalities of X. Let X_d, for $d = 0, \ldots, n$, be the corresponding relaxations produced, and X_{P_d} their respective projections as defined by (55).

Theorem 1 (Hierarchy of Relaxations). $\bar{X}_{P_d} \subseteq \bar{X}_{P_{(d-1)}} \quad \forall\, d = 1, \ldots, n.$

Proof. It is sufficient to show that any constraint $\alpha z - \beta \geq 0$ of $\bar{X}_{(d-1)}$, where $z \equiv (x, y, w, v)$, can be obtained by surrogating appropriate constraints of $\bar{X}_d$ using nonnegative multipliers. By Remark 4, we have inherent in $\bar{X}_d$, or implied by the constraints of $\bar{X}_d$, the set of inequalities

$$\{(\alpha z - \beta) \cdot (g_i x - g_{0i})\}_L \geq 0 \quad \forall\, i = 1, \ldots, p, \tag{86}$$

where the operation in (86) is defined in (78). Using (78), (77), (81), and (82), we have for any $t \in \{1, \ldots, n\}$,

$$\begin{aligned} \sum_{i=1}^{p} (\lambda_i^t + \pi_i^t)\, [(\alpha z - \beta) \cdot (g_i x - g_{0i})]_L &= [(\alpha z - \beta) x_t]_L + [(\alpha z - \beta)(1 - x_t)]_L \\ &= (\alpha z - \beta). \end{aligned}$$

Hence, (86) implies that $(\alpha z - \beta) \geq 0$, and this completes the proof. ∎

Denoting $\bar{X}_{P_0} \equiv \bar{X}_0$ as the linear programming relaxation obtained from (79) by dropping the requirement of x being binary valued, we have the following result.

Theorem 2 (Dominance of RLT1 over RLT0). $\bar{X}_d \subseteq X_d \quad \forall\, d = 0, 1, \ldots, n,$ and so, $\bar{X}_{P_d} \subseteq X_{P_d} \quad \forall\, d = 0, 1, \ldots, n.$

Proof. It is sufficient to show that for each level $d = 0, 1, \ldots, n$, any defining constraint of X_d can be obtained as a nonnegative surrogate of constraints of $\bar{X}_d$. By virtue of (81) and (82), this is clearly true for $d = 0$. By induction, assume that this assertion is true for $(d-1)$, and consider the corresponding d^{th}–level relaxations, where $d \in \{1, \ldots, n\}$. As was mentioned in Section 2.2, the constraints of X_d are obtained by taking products of the constraints of $X_{(d-1)}$ with some x_t or $(1 - x_t)$, for some t. Consider any such constraint of X_d which, under (78), is obtained by

$$[(\alpha z - \beta) \cdot x_t]_L \geq 0, \tag{87}$$

where $z \equiv (x, y, w, v)$, and where $\alpha z \geq \beta$ is a constraint defining $X_{(d-1)}$. (The case of $(1 - x_t)$ being applied in (87) is similar.) By the induction hypothesis, there exist defining constraints of

$\bar{X}_{(d-1)}$ of the type $\gamma_k z - \mu_k \geq 0$, for $k = 1, \ldots, K$, and multipliers $\phi_k \geq 0$, for $k = 1, \ldots, K$, such that

$$\sum_{k=1}^{K} \phi_k [\gamma_k z - \mu_k] = (\alpha z - \beta) \tag{88}$$

Moreover, by Remark 2, there exist constraints of $\bar{X}_d$ of the form

$$[(\gamma_k z - \mu_k) \cdot (g_i x - g_{0i})]_L \geq 0 \qquad \forall\, i, k. \tag{89}$$

Surrogating (89) via the following multipliers, where $\lambda_i^t \forall\, i$ are defined in (81), and applying (78), (77), and (88), yields

$$\begin{aligned} 0 &\leq \sum_{k=1}^{K} \phi_k \left\{ \sum_{i=1}^{p} \lambda_i^t [(\gamma_k z - \mu_k) \cdot (g_i x - g_{0i})]_L \right\} \\ &= \sum_{k=1}^{K} \phi_k [x_t (\gamma_k z - \mu_k)]_L \\ &= [(\alpha z - \beta) x_t]_L . \end{aligned}$$

Hence, a surrogate of the constraints (89) of $\bar{X}_d$ using nonnegative multipliers, produces the constraint (87) of X_d. This completes the proof. ∎

Corollary 1. $\bar{X}_0 \equiv \bar{X}_{P_0} \supseteq \bar{X}_{P_1} \supseteq \bar{X}_{P_2} \supseteq \cdots \supseteq \bar{X}_{P_n} = \mathit{conv}\,(X)$.

Proof. The inclusions $\bar{X}_0 \equiv \bar{X}_{P_0} \supseteq \bar{X}_{P_1} \supseteq \bar{X}_{P_2} \supseteq \cdots \supseteq \bar{X}_{P_n}$ are proven in Theorem 1 above. Furthermore, since $\bar{X}_{P_n}$ is a valid relaxation, we have $\text{conv}(X) \subseteq \bar{X}_{P_n}$. Moreover, we have by Theorem 2 and (56) that $\bar{X}_{P_n} \subseteq X_{P_n} = \text{conv}\,(X)$. Hence, $\bar{X}_{P_n} = \text{conv}\,(X)$, and this completes the proof. ∎

Remark 3. Note that, as introduced in the previous chapter, one can construct a hierarchy leading to the convex hull representation in a piecewise fashion as follows. Consider a partition of $N = \{1, 2, \ldots, n\}$ into disjoint sets $N_1, \ldots, N_r$ such that $\bigcup_{i=1}^{r} N_i = N$. Suppose that the foregoing scheme **RLT1** is applied by treating x_j, $j \in N_1$, as being binary valued, and the remaining x and y variables as being continuous. Accordingly, we can construct the set

$$Z_1 = \text{conv}\,\{\bar{X}_0 \cap \{(x, y) : x_j \text{ is binary for } j \in N_1\}\}, \tag{90}$$

where $\bar{X}_0$ is the usual linear programming relaxation of the feasible region X. Therefore, every vertex of Z_1 has binary values for the variables x_j, $j \in N_1$. Note that Z_1 is the projection of the highest level relaxation $\bar{X}_{|N_1|}$ onto the space of the original variables. This process can now be repeated by treating Z_1 as the set $\bar{X}_0$, the variables in N_2 as being binary valued, and the remaining variables as being continuous. Upon the projection of the corresponding highest level relaxation, we would obtain

$$Z_2 = \text{conv}\,\{Z_1 \cap \{(x,y) : x_j \text{ binary for } j \in N_2\}\}. \tag{91}$$

Note that Z_2 is the convex hull of vertices of Z_1 at which x_j, $j \in N_2$, are also binary valued. Hence, Z_2 is the convex hull of vertices of $\bar{X}_0$ at which x_j, $j \in N_1 \cup N_2$, are binary valued. Continuing in this manner, we can produce conv(X).

It is important to point out that the intermediate projection steps performed at each stage of the foregoing iterative process are not necessary to achieve a convex hull representation after r steps. After constructing factors based on the set N_i, and constructing the corresponding highest level relaxation in accordance with Remark 1, one can simply move on to the set $N_{(i+1)}$ and apply the RLT process to this higher dimensional representation itself, under (77) and (78). In this manner, the projection of the final relaxation at step r of this process onto the space of original variables (x,y) would represent the convex hull of X. Moreover, in order to computationally use the relaxations, final or intermediate, it is not necessary to perform any projection operation. The higher dimensional representations can themselves be used as linear programming relaxations. ■

3.2. Composing RLT1 factors based on special structures

In this section, we demonstrate how the general framework of **RLT1** can be exploited in the presence of special structures, and we also provide insights into the relationships between **RLT0** and **RLT1**, by considering various illustrative examples. We begin with a simple case to show how **RLT1** subsumes **RLT0**.

Example 1. Consider the set $S = \{x : 0 \le x \le e_n\}$.

Here, the S–factors of order d include $F_d(J_1, J_2)$ for all (J_1, J_2) of order d, and additionally, they generate the following factors that can be identified as being null or redundant, and hence,

can be eliminated.

(a) *Factors that include x_t and $(1-x_t)$ for some t.* These become null factors after applying the identity $x_t^2 = x_t$.

(b) *Factors $F_{d'}(J_1, J_2)$ of order $d' < d$.* (These are produced whenever at least one variable x_t is included at least twice in the factor product, using the identity $x_t^2 = x_t$.) As proven in Sherali and Adams (1990), such factors are obtainable via surrogates of factors $F_d(J_1, J_2)$ of order d.

Hence, by Remark 2, the only factors we need to employ in the multiplication process for **RLT1** are the usual **RLT0** factors $F_d(J_1, J_2)$ of order d. This yields **RLT1** $\equiv$ **RLT0** for this example. ■

Example 2. Consider the set $S = \{x : e_n \cdot x \leq 1,\ x \geq 0\}$.

The S–factors of various orders for this particular set can be derived as follows:

(a) *S–factors at level 1.* These factors are directly obtained from the set S as $(1 - e_n \cdot x) \geq 0$, and $x_j \geq 0 \quad \forall j = 1, \ldots, n$.

(b) *S–factors at level 2.* The linearization operation

$$(x_t(1 - e_n \cdot x))_L \geq 0 \tag{92}$$

produces an expression

$$\sum_{j \neq t} w_{(tj)} \leq 0 \qquad \forall t = 1, \ldots, n, \tag{93}$$

where $w_{(tj)} \equiv w_{tj}$ if $t < j$, and $w_{(tj)} \equiv w_{jt}$ otherwise. Moreover, via the pairwise products of x_j and x_t, $j \neq t$, we obtain factors of the type

$$(x_t x_j) \geq 0 \quad \forall j \neq t, \text{ or } w_{(tj)} \geq 0 \quad \forall j \neq t. \tag{94}$$

Equations (93) and (94) imply that

$$w_{(tj)} = 0 \quad \forall j \neq t. \tag{95}$$

Consequently, under (95), the only S–factors of order 2 that survive such a cancellation are self–product factors of the type $(x_t x_t) \geq 0 \quad \forall t$, and $(1 - e_n \cdot x) \cdot (1 - e_n \cdot x) \geq 0$. These yield the

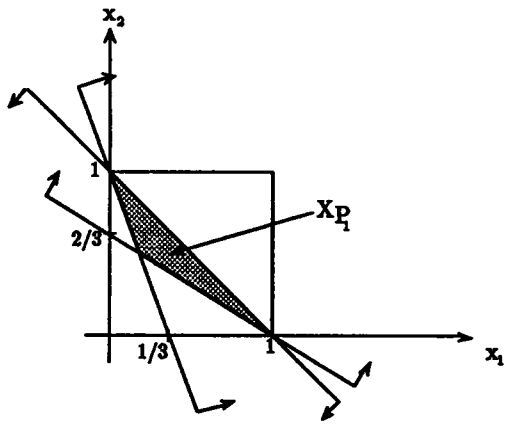

FIGURE 8. The first level relaxation using **RLT0**.

same factors as at level 1, upon using $x_t^2 = x_t \, \forall \, t$ along with (95). Hence, by Remarks 2 and 3, we only need to use the factors

$$(1 - e_n \cdot x) \geq 0 \qquad \text{and} \qquad x_j \geq 0, \quad j = 1, \ldots, n$$

to construct the equivalent set $\bar{X}_2$. Notice that $\bar{X}_2 \equiv \bar{X}_1$, and this equivalence relation continues through all levels of relaxations up to $\bar{X}_n$. Hence, the first level relaxation itself produces the convex hull representation in this case. Balas et al. (1994) have also recently demonstrated this for the special case of the maximum clique problem, and this result is also evident in Sherali and Lee (1992) (see the comment following Example 6 in Section 3). ■

There are two insightful points worthy of note in the context of this example. First, although **RLT0** recognizes that (95) holds true at each relaxation level, it may not produce the convex hull representation at the first level as does **RLT1**. For example, let

$$X = \{(x_1, x_2) \, : \, 6x_1 + 3x_2 \geq 2, \; x_1 + x_2 \leq 1, \; x \text{ binary} \},$$

and consider the generation of the first level **RLT0** relaxation. Note that the factors used in this context are x_j and $(1 - x_j)$ for $j = 1, 2$. Examining the product of $x_1 + x_2 \leq 1$ with x_1 yields $w_{12} \leq 0$, which together with $w_{12} \equiv [x_1 x_2]_L \geq 0$ yields $w_{12} \equiv 0$. Other products of the factors x_j and $(1 - x_j)$, $j = 1, 2$, with $x_1 + x_2 \leq 1$ and $0 \leq x \leq e$ simply reproduce these same latter constraints. Examining the products of $6x_1 + 3x_2 \geq 2$ with these first level factors

yields nonredundant inequalities when the factors $(1-x_1)$ and $(1-x_2)$ are used, generating the constraints $2x_1+3x_2 \geq 2$ and $3x_1+x_2 \geq 1$, respectively. Hence, we obtain the first level relaxation (directly in projected form in this case) as

$$X_{P_1} = \{(x_1, x_2) \ : \ 2x_1 + 3x_2 \geq 2, \ 3x_1 + x_2 \geq 1, \ x_1 + x_2 \leq 1, \ x \geq 0\}.$$

Figure 8 depicts the region X_{P_1}. However, by Example 2, note that $\bar{X}_{P_1} \equiv \text{conv}(X) \equiv \{x \ : \ x_1 + x_2 = 1, \ x \geq 0\}$ is a strict subset of X_{P_1}.

A second point to note is that we could have written the generalized upper bounding inequality $e_n \cdot x \leq 1$ as an equality $e_n \cdot x + x_{n+1} = 1$ by introducing a slack variable x_{n+1}, and then *recognizing the binariness of this slack variable*, we could have used this variable in composing the factor products of **RLT0**. Although this would have produced the same relaxation as with **RLT1**, the process would have generated several more redundant constraints while applying the factors x_j and $(1-x_j)$ for $j = 1, \ldots, n+1$, to the constraints, as opposed to the fewer factors $(1 - e_n \cdot x)$ and x_j, $j = 1, \ldots, n$, as needed by **RLT1**. (See Remark 4 below for a further simplification in size upon exploiting the structure of constraints produced in the presence of equality factors.) However, in more general cases of the set S, such as the one described in Example 9 in Section 5, a transformation that yields the same representation using **RLT0** as with **RLT1** may not be accessible.

Remark 4 (Equality constraint factors). If $S = \{x : \ e_n \cdot x = 1, \ x \geq 0\}$, for example, then preliminarily, by treating $e_n \cdot x = 1$ as two inequalities $e_n \cdot x \leq 1$ and $e_n \cdot x \geq 1$, we see that conv (X) could be generated as in Example 2 by using the factors $(1 - e_n \cdot x) = 0$ and $x_j \geq 0$, $j = 1, \ldots, n$, where the linearized product of an inequality with $(1 - e_n \cdot x)$ produces an equality constraint. However, given the latter fact, note that in applying **RLT1** in the presence of equality constrained factors, in general, it is only necessary to multiply such equality factors simply with each x and y variable alone. The product with any other expression in x and y can be composed using these resulting products. Moreover, since the products with the x–variables are already being generated via other **RLT1** constraints by virtue of the corresponding defining equality constraints of S already being included within X, only products using y variables are necessary. Consequently, in this example, the equality factor $(1 - e_n \cdot x) = 0$ would be used to

multiply only the $y \geq 0$ constraints in obtaining the required convex hull representation.

Furthermore, if X contains equality structural constraints, then these can be treated as in Sherali and Adams (1989). That is, at level d, these equality constraints would simply need to be multiplied by the factors $F_p(J, \emptyset)$ for $J \subseteq N$, $p = |J| = 0, 1, \ldots, d$. Naturally, factors $F_p(J, \emptyset)$ that are known to be zeros, *i.e.* , any such factor for which we know that no feasible solution exists that has $x_j = 1 \; \forall \; j \in J$, need not be used in constructing these product constraints. (Sherali and Adams accordingly suggest that in some cases, it might be useful to exploit this property by representing the structural inequality constraints as equalities via the introduction of nonnegative slack variables.) ■

Example 3. The analysis in Example 2 naturally extends to the case of generalized upper bounding (GUB), or multiple choice constraints. Here, we have,

$$S = \{x : \sum_{j \in N_i} x_j \leq 1, \quad \forall \, i \in M = \{1, \ldots, m\}, \; x \geq 0\}, \tag{96}$$

where $\bigcup_{i \in M} N_i \equiv N \equiv \{1, \ldots, n\}$. Problems possessing this particular special structure arise in various settings including maximum cardinality node packing, set packing, capital budgeting, and menu planning problems among others. (See Wolsey (1990) and Sherali and Lee (1992) for some recent polyhedral results related to such GUB constrained problems.)

In the spirit of Example 2 and Remark 3, we can construct a complete convex hull representation in the following manner. Arbitrarily, consider the variables of one of the GUB constraints, say x_j, $j \in N_1$, to be binary, treat the remaining variables as being continuous, and identify, as in Example 2, the appropriate first-order S-factors associated with this GUB constraint. Applying these factors using **RLT1** at the first level, produces the convex hull of feasible solutions for which x_j, $j \in N_1$, are restricted to be binary valued. Denote this set as $\text{conv}(X(N_1))$. We next repeat the process using N_2, considering the variables x_j, $j \in N_2$, to be binary, along with those of N_1. Applying the factors $x_j \geq 0$, $j \in N_2$, and $\left(1 - \sum_{j \in N_2} x_j\right) \geq 0$ to the foregoing relaxation, produces the set $\text{conv}(X(N_1 \cup N_2))$, with obvious notation. Continuing in this manner, upon a complete sequential application of all GUB constraint S-factors, we recover

$$\text{conv} \left(X \left(\bigcup_{i \in M} N_i \right) \right) = \text{conv} \, (X(N)).$$

To further illustrate the intermediate relaxations produced by **RLT1** for such GUB constrained problems, consider for simplicity the set S defined by the constraints

$$S = \{x \in R^5 : \; x_1 + x_2 + x_3 \leq 1, \quad x_3 + x_4 + x_5 \leq 1, \;\text{ and }\; x \geq 0\}. \tag{97}$$

Instead of applying **RLT1** sequentially as above, one GUB set at a time, let us illustrate the usual **RLT1** process by treating the constraints defining S as in Section 2. The reader can then see the equivalence of the two processes in producing the final convex hull representation.

(a) *S–factors at level 1*: These are simply the constraints defining S.

(b) *S–factors at level 2*: As in Example 2, the pairwise products within each GUB set of variables will reproduce the factors at the first level. However, across the two GUB sets, we would produce the quadratic bound factor products x_1x_4, x_1x_5, x_2x_4, and x_2x_5 along with the following factor products, recognizing that any quadratic product involving x_3 is zero.

$$x_4 \cdot (1 - x_1 - x_2 - x_3) \geq 0 \;\text{ yielding }\; x_4 - x_1x_4 - x_2x_4 \geq 0,$$

$$x_5 \cdot (1 - x_1 - x_2 - x_3) \geq 0 \;\text{ yielding }\; x_5 - x_1x_5 - x_2x_5 \geq 0,$$

$$x_1 \cdot (1 - x_3 - x_4 - x_5) \geq 0 \;\text{ yielding }\; x_1 - x_1x_4 - x_1x_5 \geq 0,$$

$$x_2 \cdot (1 - x_3 - x_4 - x_5) \geq 0 \;\text{ yielding }\; x_2 - x_2x_4 - x_2x_5 \geq 0,$$

and

$$(1 - x_1 - x_2 - x_3) \cdot (1 - x_3 - x_4 - x_5) \geq 0 \quad \text{yielding}$$

$$1 - x_1 - x_2 - x_3 - x_4 - x_5 + x_1x_4 + x_1x_5 + x_2x_4 + x_2x_5 \geq 0.$$

These can now be applied to the constraints defining X, recognizing the terms that have been identified to be zeros.

(c) *S–factors at levels* ≥ 3: Since there are only 2 GUB sets in this example, and since any triple product of distinct factors must involve a pair of factors coming from the defining constraints corresponding to the same GUB set, and the latter product is zero, all such products must vanish. Hence, all factors at level 3, and similarly at levels 4 and 5, coincide with those at level 2. In other words, the relaxation at level 2 itself yields the convex hull representation.

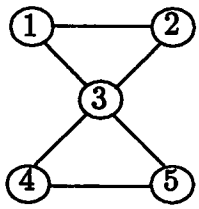

FIGURE 9. Vertex packing graph.

In general, *the level equal to the independence number of the underlying intersection graph corresponding to the GUB constraints, which simply equals the maximum number of variables that can simultaneously be 1, is sufficient to generate the convex hull representation.* ■

Remark 5. An enlightening special case of Example 3 deals with the vertex packing problem. Given a graph $G = (V, E)$ with vertex set $V = \{v_1, v_2, \ldots, v_n\}$, an edge set E connecting pairs of vertices in V, and a weight c_j associated with each vertex v_j, the vertex packing problem is to select a maximum weighted subset of vertices such that no two vertices are connected by an edge. For each $j = 1, \ldots, n$, by denoting the binary variable x_j to equal 1 if vertex j is chosen and 0 otherwise, the vertex packing problem can be stated as maximize $\{\sum_j c_j x_j : x_i + x_j \leq 1 \; \forall \; (i,j) \in E, \; x \text{ binary}\}$. The convex hull representation over any subset P of the variables can be obtained as in Example 3 by considering any clique cover of the subgraph induced by the corresponding vertices, with each set N_i corresponding to the variables defining some clique i. In fact, given a cover that has m cliques where each edge of E is included in some clique graph, *the S–factors of level m themselves define the convex hull representation*, since the packing constraints are themselves implied by these factors. To illustrate, the inequalities of (97) can be considered as a (maximum cardinality) clique cover of the vertex packing problem on the graph in Figure 9, and so, the stated S–factors of level 2 themselves define the convex hull representation. This general packing observation may have widespread applicability since, as noted by Garfinkel and Nemhauser (1973), any finite integer linear program can be reformulated as a packing problem. In Chapter 5, we illustrate the computational benefits of **RLT1** over **RLT0** in this particular context. ■

The structure afforded by GUB constraints highlights the dominating effect of **RLT1** over **RLT0**. Consider the case of constructing a $d = 1$ level **RLT1** relaxation of the MIP with two binary variables x_1 and x_2 of the form introduced earlier in Chapter 2, only this time with a single GUB constraint explicitly present:

$$
\begin{aligned}
& \min c_1x_1 + c_2x_2 + cy \\
\text{subject to: } & a_1x_1 + a_2x_2 + Ay \geq b \\
& x_1 + x_2 \leq 1 \\
& y \geq 0 \\
& x_j \in \{0,1\} \quad \forall j = 1,2.
\end{aligned} \tag{98}
$$

Instead of automatically applying the bound factors generated by the two binary variables as in RLT, we instead generate factors based on the special structure constraint set, $S = \{(x_1, x_2) : x_1 + x_2 \leq 1,\ x \geq 0\}$. In addition to the bound factors $x_1 \geq 0$ and $x_2 \geq 0$, we obtain a stronger factor $(1 - x_1 - x_2) \geq 0$ from the GUB constraint to use as a multiplier in the construction of relaxations. Thus, we have the set of **RLT1** constraints

$$
\begin{aligned}
& (a_1x_1 + a_2x_2 + Ay \geq b) \cdot (1 - x_1 - x_2) \\
& (a_1x_1 + a_2x_2 + Ay \geq b) \cdot x_1 \\
& (a_1x_1 + a_2x_2 + Ay \geq b) \cdot x_2 \\
& x_1x_2 \geq 0 \\
& x_1(1 - x_1 - x_2) \geq 0 \\
& x_2(1 - x_1 - x_2) \geq 0 \\
& y \cdot (1 - x_1 - x_2) \geq 0 \\
& y \cdot x_1 \geq 0 \\
& y \cdot x_2 \geq 0.
\end{aligned} \tag{99}
$$

Performing the indicated multiplications, and using the relationship $x_i^2 = x_i$, results in the system of inequalities given by

$$
\begin{aligned}
-a_1x_1x_2 - a_2x_1x_2 + Ay - Ayx_1 - Ayx_2 &\geq b(1 - x_1 - x_2) \\
a_1x_1 + a_2x_1x_2 + Ayx_1 &\geq bx_1 \\
a_1x_1x_2 + a_2x_2 + Ayx_2 &\geq bx_2 \\
x_1x_2 &\geq 0 \\
-x_1x_2 &\geq 0 \\
-x_1x_2 &\geq 0 \\
y - yx_1 - yx_2 &\geq 0 \\
yx_1 &\geq 0 \\
yx_2 &\geq 0.
\end{aligned}
\tag{100}
$$

Notice now the effect of the stronger multiplier. Because we are enforcing a binary condition on the special structure multipliers, all terms x_1x_2 are equal to zero, and drop from the formulation. Performing the usual substitution $v_J = y\prod_{j\in J} x_j$ yields a first level relaxation linear program

$$
\begin{aligned}
&\min c_1x_1 + c_2x_2 + cy \\
\text{subject to:}\quad &A(y - v_1 - v_2) \geq b(1 - x_1 - x_2) \\
&a_1x_1 + Av_1 \geq bx_1 \\
&a_2x_2 + Av_2 \geq bx_2 \\
&x_1 + x_2 \leq 1 \\
&y - v_1 - v_2 \geq 0 \\
&y, v_1, v_2 \geq 0 \\
&0 \leq x_j \leq 1, \qquad j = 1, 2.
\end{aligned}
\tag{101}
$$

Decomposition again reveals the essence of the **RLT1** procedure. Substituting $z = y - v_1 - v_2$ into the objective function, and projecting the problem into the original variable space by factoring

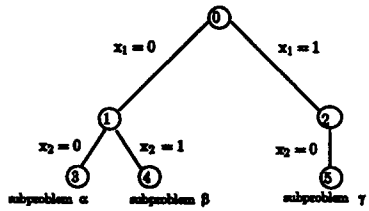

FIGURE 10. (a) Level $d = 1$ reduced node branching.

out the **RLT1** multipliers yields the decomposed problem:

$$\min_{\substack{0\le x_1\le 1\\ 0\le x_2\le 1\\ x_1+x_2\le 1}} c_1x_1 + c_2x_2 + (1 - x_1 - x_2)\underbrace{\left[\begin{array}{ll}\min\limits_{y} cy & \\ \text{st:} & Ay \ge b \\ & y \ge 0\end{array}\right]}_{P_{00}} + x_1\underbrace{\left[\begin{array}{ll}\min\limits_{y} cy & \\ \text{st:} & Ay \ge b - a_1 \\ & y \ge 0\end{array}\right]}_{P_{10}}$$

$$+ x_2\underbrace{\left[\begin{array}{ll}\min\limits_{y} cy & \\ \text{st:} & Ay \ge (b - a_2) \\ & y \ge 0\end{array}\right]}_{P_{01}}. \tag{102}$$

Again, as in RLT, depending upon the values of the objective function coefficients and the tightness of the subproblems, the minimization will select to activate one of the subproblems, and automatically yield binary decisions for x_1 and x_2. The implicit decomposition of (102) corresponds to the reduced branching tree shown in Figure 10. Comparing this level-1 structure to the level-1 and level-2 branching structures implicit in a usual application of RLT, notice that by using only the special structure factors in the first level reformulation-linearization scheme, a depth of $|J| = 2$ branchings is resolved at node 0, whereas RLT would have required a second level relaxation to achieve this result. In other words, the **RLT1** factors automatically preclude problem P_{11} from feasibility, thus accomplishing an implicit pruning of the search tree in the process.

Example 4. This example similarly points out that in the presence of variable upper bounding (VUB) types of restrictions, a further tightening of relaxations via **RLT1**, beyond that of **RLT0**, can be produced. For example, a set S might be composed as follows in a particular problem

instance:

$$S = \{x \in R^6 : 0 \le x_1 \le x_2 \le x_3 \le 1,\ 0 \le x_4 \le x_5 \le 1,\ 0 \le x_6 \le 1\}.$$

The first level factors for this instance are given by $x_1 \ge 0$, $x_2 - x_1 \ge 0$, $x_3 - x_2 \ge 0$, $1 - x_3 \ge 0$, $x_4 \ge 0$, $x_5 - x_4 \ge 0$, $1 - x_5 \ge 0$, $x_6 \ge 0$, and $1 - x_6 \ge 0$. These are not only *fewer in number* than the factors proposed by **RLT0**, but they also *yield tighter constraints* as they imply the **RLT0** factors. For $d \in \{1, \ldots, 6\}$, taking these factors d at a time, including self–products, and simplifying these factors as per Remark 2, would produce the relaxation $\bar{X}_d$. ■

It is interesting to note in this connection that the VUB constraints of the type $0 \le x_1 \le x_2 \le \cdots \le x_k \le 1$ as used in this example, can be equivalently transformed into a GUB constraint via the substitution $z_j = x_j - x_{j-1}$ for $j = 1, \ldots, k$, where $x_0 \equiv 0$. The inverse transformation yields $x_j = \sum_{t=1}^{j} z_t$ for $j = 1, \ldots, k$, thereby producing the equivalent representation $z_1 + z_2 + \cdots + z_k \le 1$, $z \ge 0$. Under this transformation, and imposing binary restrictions on all the z-variables, the reformulation strategies described for the previous two examples can be employed. However, note that the process of applying **RLT0** to the original or to the transformed problem can produce different representations. To illustrate this insight, suppose that

$$X = \{(x_1, x_2) : -6x_1 + 3x_2 \le 1,\ 0 \le x_1 \le x_2 \le 1,\ x \text{ binary }\}.$$

The convex hull of feasible solutions is given by $0 \le x_1 = x_2 \le 1$ (see Figure 11a). This representation is produced by **RLT1**, where the relevant constraint $x_1 \ge x_2$ which yields $x_1 = x_2$ is obtained by noting that the factor products $[x_1(x_2 - x_1)]_L \ge 0$ and $[x_1(1 - x_2)]_L \ge 0$ respectively give $w_{12} \ge x_1$ and $w_{12} \le x_1$, or that $w_{12} = x_1$. This together with the constraint $[(x_2 - x_1)(1 + 6x_1 - 3x_2)]_L \ge 0$ yields $-2(x_2 - x_1) \ge 0$, or that $x_1 \ge x_2$.

On the other hand, constructing **RLT0** at level 1 by applying the factors x_j and $(1 - x_j)$, $j = 1, 2$, to all the constraints, produces the relaxation (directly in projected form)

$$X_{P_1} = \{(x_1, x_2) : 2x_1 - 3x_2 \ge -1,\ 3x_1 \ge x_2,\ 0 \le x_1 \le x_2 \le 1\},$$

where $w_{12} = x_1$ is produced as with **RLT1**, and where the first two constraints defining X_{P_1} result from the product constraints $[(1 + 6x_1 - 3x_2)(1 - x_1)]_L \ge 0$ and $[(1 + 6x_1 - 3x_2)x_2]_L \ge 0$, respectively. Figure 11(a) depicts the region defined by this relaxation.

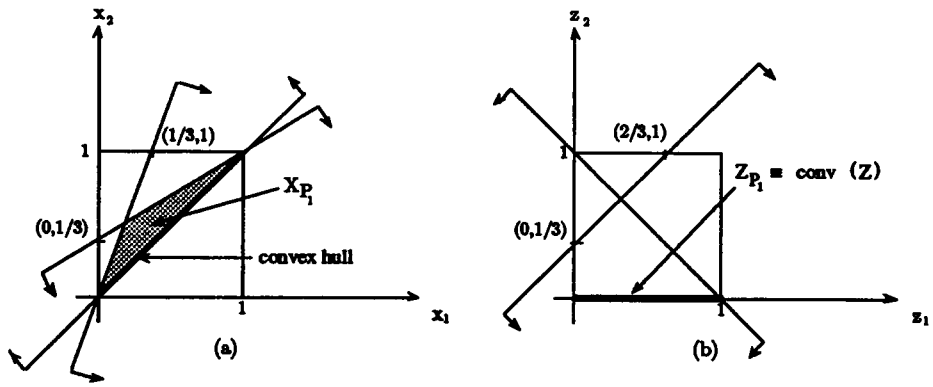

FIGURE 11. Depiction of the first level relationships using **RLT0**.

However, if we were to apply the transformation $z_1 = x_1$, $z_2 = x_2 - x_1$ to X, where the inverse transformation is given by $x_1 = z_1$ and $x_2 = z_1 + z_2$, the problem representation in z-space becomes

$$Z = \{(z_1, z_2) \ : \ -3z_1 + 3z_2 \leq 1, \ z_1 + z_2 \leq 1, \ z \text{ binary } \},$$

where the *binariness on z_2 has been additionally recognized.* Figure 11(b) illustrates that the set conv(Z) is given by the constraints $0 \leq z_1 \leq 1$, $z_2 = 0$. The relevant constraint $z_2 = 0$ is produced by **RLT0** via $z_2 \geq 0$ and the first level product constraint $[(1 + 3z_1 - 3z_2)z_2]_L \geq 0$ which yields $-2z_2 \geq 0$, where $[z_1 z_2]_L \equiv 0$. Hence, for this transformed problem, **RLT0** produces the first level relaxation $Z_{P_1} =$ conv(Z), while we had $X_{P_1} \supset$ conv(X) for the original problem. However, as with the illustration of Example 2 (treating that as the transformed x-variable problem), we could have also obtained $Z_{P_1} \supset$ conv(Z), whereas **RLT1** would have produced the convex hull representation at level one in either case.

Example 5. In this example, we illustrate how one can exploit problem sparsity. Suppose that in some 0–1 mixed–integer problem, we have the knapsack constraint (either inherent in the problem or implied by it) given by $2x_1 + x_2 + 2x_3 \geq 3$. The facets of the convex hull of $\{(x_1, x_2, x_3) \ : \ 2x_1 + x_2 + 2x_3 \geq 3, \ 0 \leq x_i \leq 1, \ i = 1, 2, 3\}$ can be readily obtained as $\{x_1 + x_2 + x_3 \geq 2, \ x_1 \leq 1, \ x_2 \leq 1, \ x_3 \leq 1\}$. Similarly, another knapsack constraint might be of the type $x_4 + 2x_5 + 2x_6 \leq 2$, and the corresponding facets of the convex hull of feasible 0–1 solutions can be obtained as $\{x_4 + x_5 + x_6 \leq 1, \ x_4 \geq 0, \ x_5 \geq 0, \ x_6 \geq 0\}$. The set S

can now be composed of these two sets of facets, along with other similar constraints involving the remaining variables on which binariness is being enforced, including perhaps, simple bound constraint factors. Note that in order to generate valid tighter relaxations, we can simply enforce binariness on variables that fractionate in the original linear programming relaxation in the present framework. Furthermore, entire convex hull representations of underlying knapsack polytopes are not necessary—simply, the condition (80) needs to be satisfied, perhaps by explicitly including simple bounding constraints. This extends the strategy of Crowder et al. (1983) in using facets obtained as liftings of minimal covers from knapsack constraints within this framework, in order to generate tighter relaxations. ■

3.3. An alternative partial application of RLT1

In this section, we discuss an alternative procedure based on the use of a partial set of S–factors within the reformulation–linearization scheme **RLT1**, in cases where the set S explicitly includes the constraints $0 \leq x \leq e_n$. Note that Example 1 of Section 2 is a special case of this type in which S is comprised only of the constraints $0 \leq x \leq e_n$. As shown there, in this case, we simply obtain **RLT1** $\equiv$ **RLT0**.

In more general cases of this type, suppose that the set S includes $0 \leq x \leq e_n$, and additionally, contains other defining constraints of X. These constraints might be either present in the original model formulation itself, or may be valid implied inequalities generated therefrom and hence included in the model. The suggested modification of **RLT1** generates a relaxation $\hat{X}_d$ at any level $d \in \{1, \ldots, n\}$ as follows, and is denoted by **RLT2**.

(a) **Reformulation.** Generate all the bound–factors $F_d(J_1, J_2)$ of order d. Additionally, generate any subset of the other possible S–factors, and use these to identify and eliminate any null terms from the factors $F_d(J_1, J_2)$ of order d. Compose a set of factors comprised of the latter reduced set of bound factor products, plus any additional S–factors as desired, and multiply each constraint of X using only these factors. Use the identity $x_j^2 = x_j \forall\, j \in \{1, \ldots, n\}$ on the factors as well as on the reformulated problem.

(b) **Linearization.** Linearize the resulting polynomial program by using the substitution defined in (54). This produces the d^{th}–level relaxation $\hat{X}_d$.

Clearly, for each $d = 1, \ldots, n$, $\hat{X}_d$ yields a valid higher dimensional relaxation. As before, denote the projection of $\hat{X}_d$ onto the space of the original variables as $\hat{X}_{P_d}$, for $d = 1, \ldots, n$, and consider the following result.

Theorem 3. For each $d = 1, \ldots, n$, we have $\hat{X}_d \subseteq X_d$, and so, $\hat{X}_{P_n} = \mathit{conv}\,(X)$.

Proof. Note that among other constraints, $\hat{X}_d$ also includes the constraints of X_d with the added restriction that certain variables in (54) or certain linearized polynomial expressions are zeros. Hence, $\hat{X}_d \subseteq X_d$. Moreover, by (56), we have that $\text{conv}(X) \subseteq \hat{X}_{P_n} \subseteq X_{P_n} = \text{conv}\,(X)$, and so, $\hat{X}_{P_n} = \text{conv}\,(X)$. This completes the proof. ∎

Observe that **RLT2** essentially applies the Sherali–Adams reformulation–linearization scheme **RLT0** with the added recognition that the inherent special structures might reveal that certain polynomial terms or expressions are zeros. Additionally, it enhances this by using any subset of additional S–factors that can be employed in an application of **RLT1**. Hence, **RLT2** provides a continuum of strategies that can be applied between **RLT0** and **RLT1**. Moreover, by insuring that factors included at level d imply those at level $d-1$, a hierarchy of relaxations are generated, which as Theorem 3 asserts, leads to the convex hull representation.

Example 6. For the sake of illustration, consider a situation in which $S = \{x \in R^4 : x_1 + x_2 + x_3 + x_4 = 2, 0 \leq x \leq e_4\}$. The following factors are derived that can be applied as in Remark 4, noting the equality constraint defining S.

(a) *Level 1 factors*: $x_j \geq 0$ and $(1 - x_j) \geq 0$, $j = 1, \ldots, 4$, and optionally, $(e_4 \cdot x - 2) = 0$ (to be multiplied by each y variable alone as in Remark 4).

(b) *Level 2 factors*: Bound factors of order 2 given by $\{x_i x_j,\ (1 - x_i)x_j,\ x_i(1 - x_j)$, and $(1 - x_i)(1 - x_j)\ \forall\, 1 \leq i < j \leq 4\}$, and optionally, any factors (to be used as in Remark 4) from the set $\{x_i - \sum_{j \neq i} x_i x_j = 0 \quad \forall\, i = 1, \ldots, 4$ obtained by multiplying $e_4 \cdot x = 2$ by each x_i, $i = 1, \ldots, 4$, and $(e_4 \cdot x - 2) = 0$ itself, obtained from $(e_4 \cdot x - 2)^2 = 0$ upon using $\sum_{j \neq i} x_i x_j = x_i\ \forall\, i\}$.

(c) *Level d factors*, $d = 3, 4$: Bound factors $F_d(J_1, J_2) \geq 0$ of order d, with the additional restriction that all 3^{rd} and 4^{th} order terms are zeros, plus optionally, factors from the optional set at level 2. Note that the valid implication of polynomial terms of order 3 being zero, for example, is obtained through the RLT process by multiplying $x_i - \sum_{j \neq i} x_i x_j = 0$ with x_k, for each i, k, $i \neq k$. This

gives $\sum_{j \neq i,k} x_i x_j x_k = 0$, which, by the nonnegativity of each triple product term, implies that $x_i x_j x_k = 0 \quad \forall\, i \neq j \neq k$. ∎

In a likewise fashion, for set partitioning problems, for example, any quadratic (or higher order) products of variables that involve a pair of variables that appear together in any constraint are zeros. More generally, any variable product term that contains variables which cannot simultaneously take on a value of 1 in any feasible solution can be restricted to zero. Sherali and Lee (1992) use this structure to present a specialization of Sherali and Adams (1990) to derive explicit reduced level d representations in their analysis of set partitioning problems. In particular, they show that the relaxation at level d needs to be constructed using products with only nonzero factors $F_p(J, \emptyset)$, for $J \subseteq N$, $p = |J| = 0, 1, \ldots, d$ as explained in Remark 4, and moreover, the constraints $F_\delta(J_1, J_2) \geq 0 \,\forall\, (J_1, J_2)$ of order $\delta = \min\{d+1, n\}$ produced by the regular **RLT0** process can be replaced simply by $F_d(J, \emptyset) \geq 0 \,\forall\, |J| = d$, $d = 1, \ldots, \delta$. (This is actually valid for any equality constrained problem in which $x_j \leq 1 \,\forall\, j$ is implied by the structural equality constraints along with $x \geq 0$.) Furthermore, they show that at the level equal to the independence number of the underlying intersection graph, the convex hull representation is obtained.

In the example that follows, we use the quadratic assignment problem to partially illustrate the foregoing idea.

Example 7. Consider the quadratic assignment problem of size m given by

$$\min\{\sum_{i=1}^{m}\sum_{j=1}^{m}\sum_{k=1}^{m}\sum_{l=1}^{m} c_{ijkl} x_{ij} x_{kl}\}$$

$$\text{subject to:} \quad \sum_{j=1}^{m} x_{ij} = 1 \qquad \forall\, i = 1, \ldots, m,$$

$$\sum_{i=1}^{m} x_{ij} = 1 \qquad \forall\, j = 1, \ldots, m,$$

$$x_{ij} \geq 0 \qquad \forall\, (i, j),$$

$$x \in \{0, 1\}.$$

By the above discussion, the level $d \geq 1$ relaxation can be obtained by computing products (including those with $x \geq 0$) using the factors $F_p(J, \emptyset)$, $p \leq d$, where the set J denotes the set

of p variables that form a partial assignment solution. Of course, at the level m, relaxation the convex hull is obtained. ■

In addition, the more general framework of **RLT2** can be utilized in deriving improved relaxations for solving set partitioning, packing, or covering problems, the 3–D assignment problem, the quadratic assignment problem (QAP), the traveling salesman problem (TSP), and other interesting combinatorial optimization problems having exploitable special structures.

3.4. Conditional logic strengthened RLT1 Constraints

In all of the foregoing discussion, depending on the structure of the problem, there is another idea that we can exploit to even further tighten the RLT constraints that are generated. To introduce the basic concept involved, for simplicity, consider the following first level RLT constraint that is generated by multiplying a factor $(\alpha x - \beta) \geq 0$ with a constraint $(\gamma x - \delta) \geq 0$, where x is supposed to be binary valued, and where the data is all–integer. (Similar extensions can be developed for mixed–integer constraints, as well as for higher–order RLT constraints in which some factor is being applied to some other valid constraint/factor product of order greater than one.)

$$\{(\alpha x - \beta)(\gamma x - \delta)\}_L \geq 0 \tag{102}$$

Observe that if $\alpha x = \beta$, then $(\alpha x - \beta)(\gamma x - \delta) \geq 0$ is valid regardless of the validity of $\gamma x \geq \delta$. Otherwise, we must have $\alpha x \geq \beta + 1$ (or possibly greater than $\beta + 1$, if the structure of $\alpha x \geq \beta$ so permits), and we can then perform standard logical preprocessing tests (zero–one fixing, coefficient reduction, etc. — see Nemhauser and Wolsey (1988) for example) on the set of constraints $\alpha x \geq \beta + 1$, $\gamma x \geq \delta$, x binary, along with possibly other constraints, to tighten the form of $\gamma x \geq \delta$ to the form $\gamma' x \geq \delta'$. For example, if $\alpha x \geq \beta$ is of the type $(1 - x_j) \geq 0$, for some $j \in \{1, \ldots, n\}$, then the restriction $\alpha x \geq \beta + 1$, x binary, asserts that $x_j = 0$, and so, $\gamma x \geq \delta$ can be tightened under the condition that $x_j = 0$. (Similarly, in a higher–order constraint, if a factor $F_d(J_1, J_2)$ multiplies $\gamma x \geq \delta$, then the latter constraint can be tightened under conditional logical tests based on setting $x_j = 1 \, \forall \, j \in J_1$ and $x_j = 0 \, \forall \, j \in J_2$.)

Additionally, the resulting constraint $\gamma'x \geq \delta'$ can be further tightened by finding the maximum $\theta \geq 0$ for which $\gamma'x \geq \delta' + \theta$ is valid when $\alpha x \geq \beta + 1$ is imposed by considering the problem

$$\theta = -\delta' + \min\{\gamma'x : \alpha x \geq \beta + 1, \text{ any other valid inequalities }, x \text{ binary }\}. \tag{103}$$

Note that, of course, we can simply solve the continuous relaxation of (103) and use the resulting value θ after rounding it upwards (using $\theta = 0$ if this value is negative), in order to impose the following RLT constraint, in lieu of the weaker restriction (102), within the underlying problem:

$$\{(\alpha x - \beta)(\gamma'x - \delta' - \theta)\}_L \geq 0. \tag{104}$$

Observe that this also affords the opportunity to now tighten the factor $(\alpha x - \beta)$ in a similar fashion in (104), based on the valid constraint $\gamma'x \geq \delta' + \theta$.

3.4.1. RLT1 and sequential lifting. Interestingly, the sequential lifting process of Balas and Zemel (1978), for lifting a minimal cover inequality into a facet for a full-dimensional knapsack polytope defined by the constraint $\sum_{j \in N} a_j x_j \geq b$, x binary, where $0 \leq a_j \leq b \, \forall \, j \in N$, can be viewed as a consequence of the above **RLT1** approach. In fact, we have the following.

Theorem 4. The valid inequality generated at any stage of sequential lifting for a knapsack constraint is implied by the sequentially generated **RLT1** constraints in which the first level products are applied to the previous stage's lifted constraints.

Proof. At any stage, consider the partially lifted valid inequality $\sum_{j \in T} \gamma_j x_j \geq \delta$, $T \subset N$. By the standard sequential lifting process, this would next be lifted to the form

$$\sum_{j \in T} \gamma_j x_j \geq \delta + (1 - x_t)\gamma_t, \tag{105}$$

where $t \in N - T$, by solving the problem

$$\gamma_t = -\delta + \min\{\sum_{j \in T} \gamma_j x_j : \sum_{j \in N} a_j x_j \geq b, \; x_t = 0, \; x \text{ binary }\}. \tag{106}$$

Now, consider the application of the level 1 **RLT1** process to the given valid inequality $\sum_{j \in T} \gamma_j x_j \geq \delta$, and let us show that this process would imply the same lifted constraint given by (105) and

(106). To this end, consider the **RLT1** constraints generated by the bound factors $x_t \geq 0$ and $x_t \leq 1$ given by

$$\{x_t \cdot [\sum_{j \in T} \gamma_j x_j - \delta]\}_L \geq 0 \text{ and } \{(1 - x_t) \cdot [\sum_{j \in T} \gamma_j x_j - \delta]\}_L \geq 0. \tag{107}$$

If $x_t = 1$, then $(1 - x_t) \cdot [\sum_{j \in T} \gamma_j x_j - \delta] \geq 0$ is valid, regardless of the validity of $\sum_{j \in T} \gamma_j x_j \geq \delta$. Otherwise, we have $1 - x_t \geq 1$, and, using conditional logic, the resulting **RLT** constraint can be further tightened by finding the maximum $\theta \geq 0$ for which $\sum_{j \in T} \gamma_j x_j \geq \delta + \theta$ is valid when $x_t = 0$. This can be done by computing

$$\theta = -\delta + \min\{\sum_{j \in T} \gamma_j x_j : x_t = 0, \sum_{j \in N} a_j x_j \geq b, x \text{ binary } \} \tag{108}$$

which yields precisely $\theta = \gamma_t$ as given in (106). Hence, the second **RLT1** constraint in (107) can be tightened to the form

$$\{(1 - x_t) \cdot [\sum_{j \in T} \gamma_j x_j - \delta - \gamma_t]\}_L \geq 0. \tag{109}$$

Summing (109) and the first **RLT1** constraint in (107) yields

$$\sum_{j \in T} \gamma_j x_j - \delta - \gamma_t + x_t \gamma_t \geq 0,$$

which is the lifted inequality of (105) and (106). This completes the proof. ∎

Note that Theorem 4 identifies an upper bound on the order of **RLT1** bound factors. Starting with some minimal cover set T_0, one needs to go through no more than $|N - T_0|$ levels of **RLT1** relaxations to have lifted the minimal cover constraint into a facet.

Example 8. Consider the following knapsack constraint in binary variables x_1, x_2, and x_3: $2x_1 + 3x_2 + 3x_3 \geq 4$. Let us examine the **RLT1** constraint

$$\{x_3 [2x_1 + 3x_2 + 3x_3 - 4]\}_L \geq 0. \tag{110}$$

Applying the idea of Theorem 4, we can tighten (110) under the restriction that $x_3 = 1$, knowing that it is always valid when $x_3 = 0$ regardless of the nonnegativity of any expression contained in $[\cdot]$. However, when $x_3 = 1$, the given knapsack constraint becomes $2x_1 + 3x_2 \geq 1$, which by

coefficient reduction, can be tightened to $x_1 + x_2 \geq 1$. Hence, (110) can be replaced by the tighter restriction

$$\{x_3 [x_1 + x_2 - 1]\}_L \geq 0. \tag{111}$$

Similarly, consider the RLT constraint

$$\{(1 - x_3) [2x_1 + 3x_2 + 3x_3 - 4]\}_L \geq 0. \tag{112}$$

This time, imposing $(1 - x_3) \geq 1$ or $x_3 = 0$, the knapsack constraint becomes $2x_1 + 3x_2 \geq 4$ which implies via standard logical tests that $x_1 = x_2 = 1$. Hence, we can impose the *equalities*

$$\{(1 - x_3)(x_1 - 1)\}_L = 0 \quad \text{and} \quad \{(1 - x_3)(x_2 - 1)\}_L = 0 \tag{113}$$

in lieu of (112), which is now implied. Observe that in this example, the sum of the RLT constraints in (111) and (113) yield $x_1 + x_2 + x_3 \geq 2$ which happens to be a facet of the knapsack polytope $\text{conv}\{x : 2x_1 + 3x_2 + 3x_3 \geq 4,\ x \text{ binary}\}$. This facet can alternatively be obtained by lifting the minimal cover inequality $x_1 + x_2 \geq 1$. ■

Example 9. To illustrate the use of this conditional logic based tightening procedure in the context of solving an optimization problem using general S-factors, consider the following problem:

$$\text{Minimize } \{x_1 + x_2 : 6x_1 + 3x_2 \geq 2,\ x \in S,\ x \text{ binary}\}, \tag{114}$$

where $S = \{(x_1, x_2) : 2x_1 + x_2 \leq 2,\ x_1 + 2x_2 \leq 2,\ x \geq 0\}$.

Figure 12 depicts this problem graphically. The integer problem has the optimal value $v(IP) = 1$, attained at the solution $(1, 0)$ or $(0, 1)$. The ordinary LP relaxation has the optimal value $v(LP) = 1/3$, attained at the solution $(1/3, 0)$.

Next, let us consider the first level relaxation (RLT0-1) produced by **RLT0**, using the factors x_j and $(1 - x_j)$, $j = 1, 2$, on all the problem constraints. Note that $[(2 - 2x_1 - x_2)x_1]_L \geq 0$ yields $w_{12} \leq 0$, which together with $[x_1 x_2]_L \equiv w_{12} \geq 0$, gives $w_{12} = 0$. The other non-redundant constraints defining this relaxation are produced via the product constraints $[(1-x_1)(1-x_2)]_L \geq 0$, $[(6x_1 + 3x_2 - 2)(1 - x_1)]_L \geq 0$, and $[(6x_1 + 3x_2 - 2)(1 - x_2)]_L \geq 0$, which respectively yield (using $w_{12} = 0$), $x_1 + x_2 \leq 1$, $2x_1 + 3x_2 \geq 2$, and $3x_1 + x_2 \geq 1$. This gives (directly in projected form)

$$X_{P_1} = \{(x_1, x_2) : x_1 + x_2 \leq 1,\ 2x_1 + 3x_2 \geq 2,\ 3x_1 + x_2 \geq 1,\ 0 \leq x \leq e_2\}.$$

Figure 12 depicts this region. The optimal value using this relaxation is given by $v(RLT0-1) = 5/7$, attained at the solution $(1/7, 4/7)$.

On the other hand, to construct the first level relaxation (RLT1-1) produced by **RLT1**, we would employ the S-factors given by the constraints in (114). As in **RLT0**, we obtain $w_{12} = 0$, and using this, the other non-redundant constraints defining this relaxation are produced via the product constraints $[(2-2x_1-x_2)(2-x_1-2x_2)]_L \geq 0$, $[(6x_1+3x_2-2)(2-2x_1-x_2)]_L \geq 0$, and $[(6x_1+3x_2-2)(2-x_1-2x_2)]_L \geq 0$, which respectively yield, $x_1+x_2 \leq 1$, $4x_1+5x_2 \geq 4$, and $2x_1+x_2 \geq 1$. This gives (directly in projected form)

$$\bar{X}_{P_1} = \{(x_1, x_2) \; : \; x_1+x_2 \leq 1, \; 4x_1+5x_2 \geq 4, \; 2x_1+x_2 \geq 1, \; 0 \leq x \leq e_2\}.$$

Figure 12 depicts this region. Note that $\bar{X}_{P_1} \subset X_{P_1}$ and that the optimal value of this relaxation is given by $v(RLT1-1) = 5/6 > v(RLT0-1)$, and is attained at the solution $(1/6, 2/3)$.

Now, consider an enhancement of RLT1-1 using conditional logic (a similar enhancement can be exhibited for RLT0-1). Specifically, consider the product constraint $[(2-2x_1-x_2)(6x_1+3x_2-2)]_L \geq 0$ of the form (102). Imposing $(2-2x_1-x_2) \geq 1$ as for (102), i.e., $2x_1+x_2 \leq 1$ yields $x_1 = 0$ by a standard logical test. This together with $6x_1+3x_2 \geq 2$ implies that $x_2 = 1$. Hence, the tightened form of this RLT constraint is $(2-2x_1-x_2)(x_1) = 0$ and $(2-2x_1-x_2)(1-x_2) = 0$. The first constraint yields $w_{12} = 0$ (as before), while the second constraint states that $x_1+x_2 = 1$. This produces the convex hull representation, and so, this enhanced relaxation now recovers an optimal integer solution. ■

The more general framework of **RLT2** can be utilized in deriving improved relaxations for solving set partitioning, packing, or covering problems, the 3–D assignment problem, the quadratic assignment problem (QAP), the traveling salesman problem (TSP), and other interesting combinatorial optimization problems having exploitable special structures. In Chapter 4, we examine several exploitation strategies for the asymmetric traveling salesman problem (ATSP) that result in formulations that produce tightened lower bounds for linear programming relaxations of ATSP.

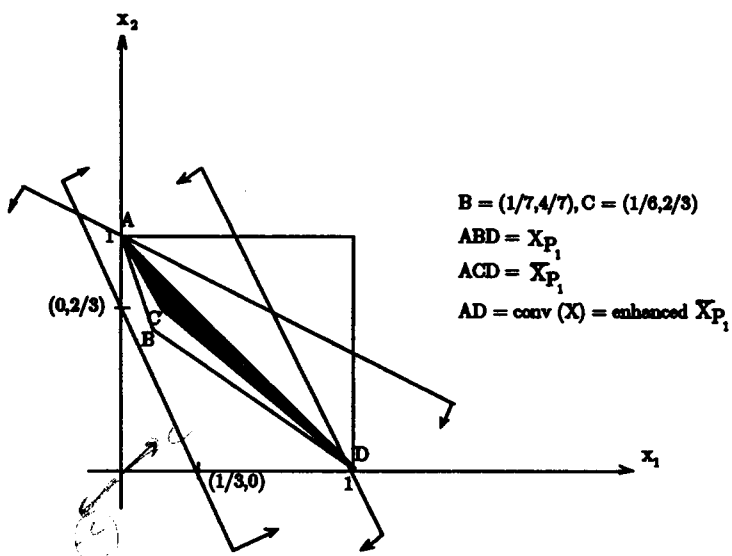

FIGURE 12. Depiction of the various first level RLT relaxations.

CHAPTER 4

TSP Reformulations

To illustrate the effectiveness of **RLT1** in tightening linear programming relaxations, we apply this approach in the present chapter to several formulations of the combinatorial optimization problem, the Traveling Salesman Problem (TSP). We begin our discussion by presenting an approach to tighten one such formulation involving the Miller-Tucker-Zemlin (1960) subtour elimination constraints. We then demonstrate a new result that a class of constraints resulting from this approach are, in fact, facetial. We next apply **RLT1** to the Miller, Tucker, and Zemlin subtour elimination constraints, composing an appropriate special-structure set that includes the new conditional bound factor products introduced in Chapter III. Lastly, utilizing the unique variable substitution afforded by RLT, we derive a new 3-index TSP formulation based on the quadratic assignment problem formulation of TSP, introducing a new set of constraints unique to the **RLT1** formulation.

4.1. The Traveling Salesman Problem

A classic statement of the traveling salesman problem can be given as follows: a salesman is required to visit n cities, indexed by $1, 2, \ldots, n$, leaving some base city, say city 1, visiting each of the other n cities exactly once, and then returning to city 1. It is required to determine such an itinerary that minimizes the total distance traveled by the salesman.

In a graph theoretical sense, this problem statement translates to the task of determining a least cost Hamiltonian circuit on a complete graph $G(V, A)$, where $V = \{v_1, v_2, \ldots, v_n\}$ is the set of vertices, and $A = \{(v_i, v_j) : v_i, v_j \in V,\ i \neq j\}$ is the set of arcs. It is commonly assumed that

the matrix of costs, $C = \{c_{ij}\}$, or distances between cities, is positive. We assume that $c_{ij} \neq c_{ji}$ possibly, so that we are explicitly addressing the *asymmetric traveling salesman problem*. All formulations of the ATSP must contain, in addition to assignment constraints, some construction that prevents *subtours* from forming in any feasible tour.

4.1.1. Dantzig, Fulkerson, and Johnson (DFJ) formulation. Although there are several valid formulations for ATSP presented in the literature (see Padberg and Sung (1992) for a survey) that we examine in the context of RLT, we initially consider the formulation credited to Dantzig, Fulkerson, and Johnson (1954) given by:

(DFJ) Minimize $\sum_i \sum_j c_{ij} x_{ij}$

subject to:

(115) $\sum_{j \neq i} x_{ij} = 1 \quad \forall \; i = 1, \ldots, n,$

(116) $\sum_{i \neq j} x_{ij} = 1 \quad \forall \; j = 1, \ldots, n,$

(117) $\sum_{i,j \in S} x_{ij} \leq |S| - 1 \quad \forall \; S \subseteq N,\ 2 \leq |S| \leq (n-1),$

(118) $x_{ij} \geq 0 \quad \forall \; i, j,$

(119) x_{ij} binary $\quad \forall \; i, j.$

The constraints generated by (115), (116) and (119) form *assignment* constraints, and insure that each city is visited exactly once in, perhaps, a collection of subtours. The constraints generated by (117) are *subtour elimination* constraints that prevent feasible assignment solutions from including subtours that do not yield a Hamiltonian circuit.

Now, let X define the set of solutions feasible to the system given by (115)–(119). It is well known that the set of feasible solutions to the linear programming relaxation given by (115)–(118), $\bar{X}$, differs from the convex hull of binary integer feasible solutions, conv(X). Thus, the solution to the linear programming relaxation of ATSP can, at best, only be expected to provide limited bounding information relative to the solution obtained when the binary nature of the

variables is enforced. There does exist some degree of commonality between the LP relaxation and the convex hull of feasible solutions that underscores the importance of linear programming relaxations in solving the ATSP. Not surprisingly, it is the tenets of polyhedral theory that provide insights into the underlying structures present in the TSP. We mention several results here to illustrate the usefulness of these insights in identifying classes of constraints that can be exploited within an algorithm such as branch-and-bound. Additionally, when incorporated into a TSP formulation, these constraints tighten the associated linear programming relaxation, yielding tighter lower bounds with which to guide LP-based branch-and-bound algorithms.

Grötschel and Padberg (1975) establish the fact that the *dimensions* of the two convex hulls are equal, *i.e.*, $\dim(\bar{X}) = \dim(\text{conv}(X)) = n^2 - 3n + 1$. Later, Grötschel and Padberg (1979) proved that the constraints in (117) can define facets of various polytopes associated with both the symmetric TSP on an undirected complete graph and the asymmetric TSP on a complete digraph in which every two distinct vertices i and j are linked by two antiparallel arcs (i, j).

Let E be a finite ground set and ψ be a set, or collection, of subsets of E. For every element $e \in E$ we can associate a variable x_e, i.e., a component of a vector $x \in R^{|E|}$ indexed by e. With every subset $F \subseteq E$ we associate an *incidence vector* of F defined as:

$$x_e^{|F|} = \begin{cases} 1 & \text{if } e \in F, \\ 0 & \text{if } e \notin F. \end{cases}$$

Thus, every subset $F \subseteq E$ corresponds to a unique 0-1 vector in $R^{|E|}$ and vice versa. Now, we can associate with $\psi \in 2^{|E|}$ the polytope P_ψ which is the convex hull of all incidence vectors of elements of ψ, i.e., $P_\psi \equiv \text{conv}\{x^{|F|} \in R^{|E|} : F \in \psi\}$. Then, every vertex of P_ψ corresponds to a set in ψ, and vice-versa. A polyhedron $P \subseteq R_+^n$ is called *monotone* if $y \in P$ and $0 \leq x \leq y$ imply $x \in P$.

Now, denote by H_n the set of all *tours* (*Hamiltonian cycles*) in the complete graph on n vertices, and let $\tilde{H}_n$ be the set of all subsets of tours, i.e., $\tilde{H}_n = \{S \subseteq E : \exists \text{ a tour } T \subseteq E \text{ with } S \subseteq T\}$. In a similar manner, let Γ_n be the set of all directed tours in the complete digraph on n vertices D_n, and let $\tilde{\Gamma}_n$ denote the set of all subsets of tours in Γ_n. Lastly, denote by $x(S)$ the sum over elements of S, i.e., $x(S) = \sum_{j \in S} x_j$. The constraints in (117) define facets specific to the TSP

polytopes under the following conditions:

(1) the subtour elimination constraints $x(A(W)) \leq |W| - 1$, where $A(W)$ denotes the arc set of the digraph $W \subseteq N$, define facets of the asymmetric TSP polytope $P_T^n \equiv \text{conv}\{x^{|T|} \in R^{|A|} : T \in \Gamma_n\}$ for $n \geq 6$ if $2 \leq |W| \leq (n-2)$;

(2) the subtour elimination constraints $x(A(W)) \leq |W| - 1$ define facets of the monotone asymmetric TSP polytope $\tilde{P}_T^n \equiv \text{conv}\{x^{|S|} \in R^{|A|} : S \subseteq \tilde{\Gamma}_n\}$ for $n \geq 3$ if and only if $2 \leq |W| \leq (n-1)$; and,

(3) the subtour elimination constraints $x(E(W)) \leq |W| - 1$, where $E(W)$ denotes the edge set of the subgraph induced by $W \subseteq N$, define facets of the monotone symmetric TSP polytope $\tilde{Q}_T^n \equiv \text{conv}\{x^{|S|} \in R^{|E|} : S \subseteq \tilde{H}_n\}$ for $n \geq 6$ if and only if $3 \leq |W| \leq (n-3)$.

It is well known that each of the aforementioned polytopes are contained within the assignment polytope for both the symmetric and asymmetric cases, since every tour of the traveling salesman problem is known to be a vertex of the assignment polytope, but not vice-versa. Additionally, it is clear that the symmetric TSP polytope $Q_T^n \equiv \text{conv}\{x^{|T|} \in R^{|E|} : T \subseteq H_n\}$ and P_T^n as defined above, are contained in their monotone counterparts.

The identification of facet defining inequalities for the convex hull representation is important in combinatorial optimization problems such as ATSP because it provides a means to generate improved formulations that facilitate the solution of larger problem instances. By adding such facet defining inequalities to a formulation of ATSP, we obtain a tighter representation of the problem in the sense of the linear programming relaxation. Ideally, we would prefer to include only facet defining inequalities in the constraint set, thereby obtaining a minimal representation of the underlying polytope. However, this is generally not possible except for a very limited class of combinatorial optimization problems. For instance, for the problem of finding a maximum–weight matching in a graph, Edmonds (1965) has prescribed a complete linear description of the polytope. Even if facet defining inequalities are not readily available, our objective still remains to tighten the constraints as much as possible.

4.1.2. Miller, Tucker, and Zemlin (MTZ) subtour elimination constraints. Miller, Tucker and Zemlin (MTZ) (1960) proposed an alternative formulation for the subtour elimination

constraints of (117). By using an additional set of free variables u_i, $i = 2, \ldots, n$, their formulation eliminates subtours via the constraints

$$u_j \geq u_i + 1 - (n-1)(1 - x_{ij}) \qquad \forall \; i \geq 2,\; j \geq 2,\; j \neq i, \tag{120}$$

$$1 \leq u_i \leq (n-1) \qquad \forall \; i = 2, \ldots, n, \tag{121}$$

$$u_1 \equiv 0. \tag{122}$$

These MTZ subtour elimination constraints, in lieu of (117), were designed to prevent subtours that do not involve the base city 1 from being generated. To illustrate, suppose that there exists an arbitrary subtour of length k given by $ST = \{i_1, \ldots, i_k\}$ where the base city $1 \notin ST$. Summing the constraints given by (120) for this particular subtour yields $\sum_{t=1}^{k} u_{i_t} \geq \sum_{t=1}^{k} u_{i_t} + k$, which cannot be true since $k \geq 2$. Additionally, every valid TSP tour remains feasible to these constraints. The values of u_i and u_j are obtained by enforcing the convention that if city i is the j^{th} city visited after the base city 1, we set $u_i = j$.

Besides working to defeat subtours, including the MTZ constraints in the ATSP formulation reduces the number of subtour elimination constraints typically generated by (117). However, Langevin, Soumis, and Desrosiers (1990), and Padberg and Sung (1992) have shown that the MTZ formulation provides a weak LP relaxation because the MTZ subtour elimination constraints (120) and (121) are not facets of the polytope of the convex hull of feasible solutions defined by (115), (116), (118)– (122).

Desrochers and Laporte (1991) strengthened the MTZ subtour elimination constraints by employing the sequential lifting technique introduced by Padberg (1973), Balas and Zemel (1978), and Zemel (1978) to indeed make these constraints facetial. Their resulting formulation is given

by

(DL) Minimize $\sum_i \sum_j c_{ij} x_{ij}$

subject to:

(123) $$\sum_{j \neq i} x_{ij} = 1 \qquad \forall \quad i = 1, \ldots, n,$$

(124) $$\sum_{i \neq j} x_{ij} = 1 \qquad \forall \quad j = 1, \ldots, n,$$

(125) $$u_j \geq (u_i + 1) - (n-1)(1 - x_{ij}) + (n-3)x_{ji} \qquad \forall \quad i, j \geq 2, \; i \neq j,$$

(126) $$u_j \geq 1 + (1 - x_{1j}) + (n-3)x_{j1} \qquad \forall \quad j = 2, \ldots, n,$$

(127) $$u_j \leq 1 + (n-3)(1 - x_{1j}) + x_{j1} \qquad \forall \quad j = 2, \ldots, n,$$

(128) x binary,

where the constraints $1 \leq u_i \leq (n-1)$ for all $i = 2, \ldots, n$ are implied by (126) and (127). (The variable u_1 is assumed to be equal to 0 as in (122) and does not appear in this formulation.) The lifted constraints of (125) were demonstrated to be facetial inequalities, based on the fact that the corresponding constraints (117) for $|S| = 2$ are facets, given $n \geq 6$. Throughout the remainder of this chapter, we will assume that $n \geq 6$, unless otherwise noted.

To demonstrate the effectiveness of this formulation in defeating subtours involving the base city 1, consider the case when $n = 6$ and two subtours exist: $ST_1 = \{1, 2, 3\}$ and $ST_2 = \{4, 5, 6\}$. Constraints (126) and (127) yield $u_2 = 1$. For $j = 3$, (126) and (127) give $u_3 \leq 5$. However, with $j = 3$ and $i = 2$, (125) yields the contradiction $5 \leq 2$, thus defeating the subtour involving the base city 1. Note that the previous MTZ constraints (120) – (122) would only have attacked the subtour ST_2, finding no direct contradiction within ST_1 itself.

4.1.3. Tightening the MTZ formulation. Desrochers and Laporte (1991) demonstrated that their formulation provides a significant computational improvement over the MTZ formulation for the symmetric TSP, but noted very little improvement in tightening the representation for the ATSP. Their computational study investigated TSPs for three distance classes: Asymmetric random (AR), Symmetric random (SR), and Symmetric Euclidean (SE). For each distance class, ten 50-city problems were generated and in each case, four values were computed:

TSP	AR	SR	SE
MTZ/ASS	1.000	1.009	1.007
DFJ/ASS	1.012	1.279	1.229
MTZ$^+$/ASS	1.006	1.274	1.211

TABLE 5. Computational results for TSPs (DL)

(1) ASS: the value of the assignment bound.

(2) MTZ: the solution value obtained by incorporating all violated MTZ constraints.

(3) MTZ$^+$: the solution obtained by incorporating all violated lifted MTZ constraints.

(4) DFJ: the solution value obtained by incorporating all DFJ constraints.

The ratios MTZ/ASS, MTZ$^+$/ASS, and DFJ/ASS were computed and averaged over the 10 problems, and are shown in Table 5. The difference between MTZ$^+$/ASS and the other two averages indicates the relative strength of the lifted MTZ contraints for the TSP.

They assert and prove the validity of (126) and (127), whereas, in fact, as we show next, these constraints are actually also facet defining inequalities.

Proposition 1. The constraints given by (126) and (127), are facet defining inequalities for the ATSP.

Proof. It is straightforward to demonstrate the validity of these constraints, and since Desrochers and Laporte accomplish this, we do not repeat their argument here. However, to demonstrate the facetial property of (126) and (127), we use a result due to Grötschel and Padberg (1975) that constraints (117) define facets of conv(X) for $|S| = 2$ and $n \geq 6$. For any ATSP solution for which $x_{1j} + x_{j1} = 1, \quad j = 2, \ldots, n$, and $|S| = 2$, constraints (117) are tight. We consider two cases: $x_{1j} = 0$ and $x_{1j} = 1$. When $x_{1j} = 0$, we have $x_{j1} = 1$ and (126) and (127) yield $u_j = (n-1)$. Similarly, when $x_{1j} = 1$ and $x_{j1} = 0$, (126) and (127) yield $u_j = 1$. Using a result due to Desrochers and Laporte (1991) that there exists a one–to–one correspondence between the polytope of (115)—(119) and the polytope of the convex hull of binary integer solutions defined by (123)—(128), the correspondence exists, in particular, for the face defined by $x_{1j} + x_{j1} = 1, \quad j = 2, \ldots, n$. Therefore, the faces defined by (126) and (127) have the same

dimension as those defined by (117) for $|S| = 2$ and $n \geq 6$. The proposition follows directly. ■

The following proposition is actually implicit in the proof of the proposition that (125)—(127) are facetial constraints. However, it is stated and proved here explicitly for completeness.

Proposition 2. The constraints given by (125)—(127) imply the DFJ subtour elimination constraints of (117) for $|S| = 2$.

Proof. The proposition is easily verified for $S = \{1, j\}$, $j \geq 2$, by multiplying (127) by -1 and adding the result to (126), yielding the relationship

$$(x_{1j} + x_{j1} - 1) \geq (n-3)(x_{1j} + x_{j1} - 1).$$

Since we have assumed that $n \geq 6$, this implies that $x_{1j} + x_{j1} \leq 1$, or that (117) holds. Now, pick two cities i and j, $i, j \geq 2$, $i \neq j$. Then, (125) yields

$$u_j \geq u_i + 1 - (n-1)(1 - x_{ij}) + (n-3)x_{ji}$$

and

$$u_i \geq u_j + 1 - (n-1)(1 - x_{ji}) + (n-3)x_{ij}.$$

Summing these two inequalities yields

$$\begin{aligned} 0 &\geq 2 - (n-1)[2 - x_{ij} - x_{ji}] + (n-3)[x_{ji} + x_{ij}] \\ &= (2n-4)[x_{ij} + x_{ji} - 1]. \end{aligned}$$

Again, since $n \geq 6$, we get $x_{ij} + x_{ji} \leq 1$, or that (117) is implied for $|S| = 2$. ■

4.2. Composing RLT1 factors for MTZ constraints

In this section, we compose special factor products that exploit the special structures of the MTZ formulation defined by (120)—(124), and (128). We begin by stating our nonlinear reformulation resulting from the first step of **RLT1** in its completeness, and then proceed to identify the specific **RLT1** constructions that support it. We then linearize this formulation, explicitly modeling the ATSP in a higher dimensional space, and algebraically simplify the linearization to reduce the actual number of constraints that are added to the original formulation.

Reformulation. Using first-order product factors of the form $g(J) = g_i x - g_{0_i}$, the complete first level ($d = 1$) **RLT1** nonlinear reformulation is given by

$$\sum_{j \neq i} x_{ij} = 1 \qquad \forall \ \ i = 1, \ldots, n, \tag{129}$$

$$\sum_{i \neq j} x_{ij} = 1 \qquad \forall \ \ j = 1, \ldots, n, \tag{130}$$

$$\sum_{j \neq i} u_i x_{ij} = u_i \qquad \forall \ \ i \geq 2, \tag{131}$$

$$\sum_{i \neq j} u_j x_{ij} = u_j \qquad \forall \ \ j \geq 2, \tag{132}$$

$$x_{ij} \cdot (u_j - 2) \geq 0 \qquad \forall \ \ i, j \geq 2, \ i \neq j, \tag{133}$$

$$x_{ji} \cdot (u_j - 1) \geq 0 \qquad \forall \ \ i, j \geq 2, \ i \neq j, \tag{134}$$

$$x_{ij} \cdot (n - 1 - u_j) \geq 0 \qquad \forall \ \ i, j \geq 2, \ i \neq j, \tag{135}$$

$$x_{ji} \cdot (n - 2 - u_j) \geq 0 \qquad \forall \ \ i, j \geq 2, \ i \neq j, \tag{136}$$

$$(u_j - 1) \cdot (1 - x_{ij} - x_{ji}) \geq 0 \qquad \forall \ \ i, j \geq 2, \ i \neq j, \tag{137}$$

$$(n - 1 - u_j) \cdot (1 - x_{ij} - x_{ji}) \geq 0 \qquad \forall \ \ i, j \geq 2, \ i \neq j, \tag{138}$$

$$(u_j - 2) \cdot (1 - x_{1j} - x_{j1}) \geq 0 \qquad \forall \ \ j \geq 2, \tag{139}$$

$$(n - 2 - u_j) \cdot (1 - x_{1j} - x_{j1}) \geq 0 \qquad \forall \ \ j \geq 2, \tag{140}$$

$$u_j x_{ij} = (u_i + 1) x_{ij} \qquad \forall \ \ i, j \geq 2, \ i \neq j, \tag{141}$$

$$u_j x_{1j} = x_{1j} \qquad \forall \ \ j \geq 2, \tag{142}$$

$$u_j x_{j1} = (n - 1) x_{j1} \qquad \forall \ \ j \geq 2, \tag{143}$$

$$1 \leq u_j \leq (n - 1) \qquad \forall \ \ j \geq 2, \tag{144}$$

$$u_1 \equiv 0 \quad \text{and } x \text{ binary}. \tag{145}$$

Constraints (131) and (132) are the result of multiplying the continuous bounds $u_i \geq 0$ and $u_j \geq 0$ by the S-factors directly obtainable from (129) and (130). Note that any constraints resulting from multiplying (129) and (130) by other bound factors of u_j, such as $(u_j - 1) \geq 0$,

are implied by (129)—(132), and hence, are not included in the formulation. Constraints (134) and (135) result from multiplying the upper and lower bounds on u_j, given by the now implied constraint $1 \leq u_j \leq (n-1)$, with the nonnegative bound factor products $x_{ij} \geq 0$ and $x_{ji} \geq 0$.

The constraints of (133) and (136) are new *conditional factor products* that result from the following reasoning. Suppose that we generate a first level **RLT1** constraint by multiplying a constraint $(\alpha u - \beta) \geq 0$ with an S-factor $(\gamma x - \delta) \geq 0$, where u is integer valued, x is binary valued, and the data is all integer:

$$\{(\alpha u - \beta)(\gamma x - \delta)\}_L \geq 0. \tag{146}$$

Notice that if $\alpha u = \beta$, then $(\alpha u - \beta)(\gamma x - \delta) \geq 0$ is valid regardless of the validity of $\gamma x \geq \delta$. Otherwise, we must have $\alpha u \geq \beta + 1$ (or possibly greater than $\beta + 1$, if the structure so permits), and we can then perform standard logical preprocessing tests (zero-one fixing, coefficient reduction, etc.– see Nemhauser and Wolsey (1988) for example) on the set of constraints $\alpha u \geq \beta + 1$, $\gamma x \geq \delta$, x binary, along with possibly other constraints, to tighten the form of $\gamma x \geq \delta$ to the form $\gamma' x \geq \delta'$. In this particular case, if $x_{ij} = 1$, for $i, j \geq 2$, $i \neq j$, we can then tighten the lower bound on u_j to $u_j \geq 2$. Similarly, if $x_{ji} = 1$ for $i, j \geq 2$, $i \neq j$, we can tighten the upper bound on u_j to $u_j \leq (n-2)$. These conditional bounds on u_j yield the implied factors $(u_j - 2) \geq 0$ and $(n - 2 - u_j) \geq 0$ that are multiplied by x_{ij} and x_{ji}, respectively, to yield (133) and (136).

Similarly, in the usual **RLT0** process, the original bound factor constraints $(1 - x_{ij}) \geq 0$ would be used to multiply the bound factor constraints (121) given by $1 \leq u_j \leq (n-1)$. However, the condition $x_{ij} + x_{ji} \leq 1$ for $i, j \geq 2$, $i \neq j$, and $x_{1j} + x_{j1} \leq 1$ for $j \geq 2$, directly indicated by the DFJ subtour elimination constraints of (117) for $|S| = 2$, imply the constraints $x_{ij} \leq 1$ under nonnegativity of the variables. These constraints can be used to compose a special-structure set S of the type noted in section 4.2, given by $S = \{x_{ij} + x_{ji} \leq 1, \forall\, i, j \geq 2,\ i \neq j,\ x_{1j} + x_{j1} \leq 1, \forall\, j \geq 2,\ x_{ij} \geq 0, \forall\, i, j,\ i \neq j\}$. The resulting **RLT1** constraints constructed by exploiting the special structure of S recover partial convex hull representations based on imposing integrality on only pairs of variables x_{ij} and x_{ji} taken two at a time. The more general case of $S = \{x : e_n \cdot x \leq 1,\ x \geq 0\}$, along with the special case of generalized upper bounding (GUB), or multiple choice constraints in which $S = \{x : \sum_{j \in N_i} x_j \leq 1, \quad \forall\, i \in M = \{1, \ldots, m\},\ x \geq 0\}$, where

$\bigcup_{i \in M} N_i \equiv N \equiv \{1, \ldots, n\}$, were shown by Sherali, et al. (1994) to possess this property. Balas, et al. (1994) recently demonstrated the ability of these type of factors to recover the convex hull at the first level of relaxation for the special case of the maximum clique problem, after imposing integrality on only a single clique. Consequently, we incorporate these stronger **RLT1** S-factors in forming the constraints (137)–(140). Additionally, we use the new conditional factor products $(u_j - 2) \geq 0$ and $(n - 2 - u_j) \geq 0$ to further tighten (139) and (140) in a manner similar to that used for (133) and (136). If $x_{1j} = 1$ or $x_{j1} = 1$, then (139) and (140) hold, regardless of the validity of $(u_j - 2) \geq 0$ or $(n - 2 - u_j) \geq 0$. If $x_{1j} + x_{j1} = 0$, then $1 < u_j < (n-1)$ holds true necessarily. Hence, we have $2 \leq u_j \leq (n-2)$, which, in turn, produces the factors used.

Constraints (141)—(143) represent a set of tight **RLT1** equalities involving the variables u_i and u_j under the desirable conditions of an ATSP formulation that would prevent subtours. The specific construction of these constraints results from the following two alternative viewpoints. First, from a logical approach, assuming that the base city is node 1, we can specify the set of equality constraints:

$$
\begin{aligned}
& u_j = (u_i + 1) \quad \text{if} \quad x_{ij} = 1, \text{ i.e. } x_{ij}u_j = x_{ij}(u_i + 1) \text{ for } i, j \geq 2,\ i \neq j, \\
& u_j = 1 \qquad \text{if} \quad x_{1j} = 1, \text{ i.e. } x_{1j}u_j = x_{1j} \qquad \text{for } j \geq 2, \\
& u_j = (n-1) \ \text{if} \quad x_{j1} = 1, \text{ i.e. } x_{j1}u_j = x_{j1}(n-1) \ \text{for } j \geq 2,
\end{aligned} \tag{147}
$$

where $1 \leq u_j \leq (n-1)$ for all $j \geq 2$. Notice that these constraints yield a desired contradiction when subtours exist even when the variables x_{ij} assume positive fractional values around subtours.

The second viewpoint is oriented more towards **RLT1**, and is similar to the derivation of the other constraints. Consider the multiplication of (120) with the bound factor $x_{ij} \geq 0$. Using conditional logic that when $x_{ij} = 1$, we must have $u_j = u_i + 1$, produces (141) as an RLT constraint. Multiplying (120) by the factor $(1 - x_{ij}) \geq 0$ upon using (141) reproduces (120) itself. In fact, multiplying (120) by the stronger S-factor $(1 - x_{ij} - x_{ji}) \geq 0$ produces the facetial constraint of Desrochers and Laporte.

To see this, note that the multiplication of (120) by the S-factor $(1 - x_{ij} - x_{ji})$ yields

$$
\begin{aligned}
u_j - u_j x_{ij} - u_j x_{ji} \geq\ & (u_i + 1)(1 - x_{ij} - x_{ji}) - (n-1)(1 - x_{ij}) \\
& + (n-1)(1 - x_{ij})x_{ij} + (n-1)(1 - x_{ij})x_{ji}.
\end{aligned} \tag{148}
$$

Using the RLT identity $x_{ij}^2 = x_{ij}$ and that $x_{ij}x_{ji} = 0$ since $x_{ij} + x_{ji} \leq 1$, (148) becomes

$$(u_j - u_i) \geq (u_j - u_i)x_{ij} + (u_j - u_i)x_{ji} + (1 - x_{ij})(2 - n) + nx_{ji} - 2x_{ji}. \tag{149}$$

From (141), we have the conditions

$$(u_j - u_i)x_{ij} = x_{ij} \qquad \text{and} \qquad (u_j - u_i)x_{ji} = -x_{ji}, \tag{150}$$

which, when substituted into (149) yields (125). As we shall show in the sequel, (125) is itself already implied by the reformulation we have constructed so far, and so these constructions need not be included above (see Proposition 3).

It is interesting to note that the facetial constraints in (125) that are obtained by Desrochers and Laporte through lifting are directly obtainable via **RLT1** via Theorem 4. Again, Theorem 4 identifies an upper bound on the order of **RLT1** bound factors used to multiply the MTZ constraints. Starting with some minimal cover set T_0, one needs to go through no more than $|N - T_0|$ levels of **RLT1** relaxations to have lifted the minimal cover constraint into a facet.

Secondly, as a consequence of Theorem 4, since the constraints of (125) are obtained via a single lifting involving the variable x_{ji}, we can recover (125) directly from the first level **RLT1** constraints generated via (120). When the usual MTZ constraint (120) is multiplied by the simple bound factor $(1 - x_{ji}) \geq 0$, characteristic of **RLT1**, we get

$$[u_j - (u_i + 1) + (n - 1)(1 - x_{ij})] \cdot (1 - x_{ji}) \geq 0, \tag{151}$$

or, using the fact that $x_{ij}x_{ji} = 0$ since $x_{ij} + x_{ji} \leq 1$, we get

$$(u_j - u_i) \geq (u_j - u_i)x_{ji} + (n - 2)x_{ji} + (2 - n) + (n - 1)x_{ij}. \tag{152}$$

Similarly, multiplying (120) by x_{ij} and using the conditional logic that, in this case, when $x_{ji} = 1$ we have $u_i = u_j + 1$, we obtain the **RLT1** constraint

$$x_{ji}(u_j - u_i + 1) = 0. \tag{153}$$

Using (153) to substitute $x_{ji}(u_j - u_i) = -x_{ji}$ into the right-hand side of (152) yields the constraint

$$u_j \geq (u_i + 1) - (n - 1)(1 - x_{ij}) + (n - 3)x_{ji}, \tag{154}$$

which is constraint (125).

Linearization Phase. After performing the appropriate multiplications indicated in the constraints (133)—(143), we next linearize the resulting formulation in accordance with (54) by using the variable substitutions $y_{ij} = u_i x_{ij}$, and $z_{ij} = u_j x_{ij}$ for all $i, j \geq 2$, $i \neq j$. Additional simplification is done by employing the relationships $u_j x_{1j} = x_{1j}$ and $u_j x_{j1} = (n-1)x_{j1}$ explicitly stated in (142) and (143).

Thus, we arrive at the linear set of constraints:

$$\sum_{j \neq i} x_{ij} = 1 \qquad \forall \ i, \tag{155}$$

$$\sum_{i \neq j} x_{ij} = 1 \qquad \forall \ j, \tag{156}$$

$$\sum_{\substack{j \neq i, \\ j \neq 1}} y_{ij} + (n-1)x_{i1} = u_i \qquad \forall \ i \geq 2, \tag{157}$$

$$\sum_{\substack{i \neq j, \\ i \neq 1}} z_{ij} + x_{1j} = u_j \qquad \forall \ j \geq 2, \tag{158}$$

$$z_{ij} \geq 2x_{ij} \quad \text{and} \quad y_{ij} \geq x_{ij} \qquad \forall \ i, j \geq 2, \ i \neq j, \tag{159}$$

$$z_{ij} \leq (n-1)x_{ij} \quad \text{and} \quad y_{ij} \leq (n-2)x_{ij} \qquad \forall \ i, j \geq 2, \ i \neq j, \tag{160}$$

$$z_{ij} + y_{ji} \leq (u_j + x_{ij} + x_{ji} - 1) \qquad \forall \ i, j \geq 2, \ i \neq j, \tag{161}$$

$$z_{ij} + y_{ji} \geq u_j + (n-1)(x_{ij} + x_{ji} - 1) \qquad \forall \ i, j \geq 2, \ i \neq j, \tag{162}$$

$$u_j \geq 1 + (n-3)x_{j1} + (1 - x_{1j}) \qquad \forall \ j \geq 2, \tag{163}$$

$$u_j \leq (n-2) - (n-3)x_{1j} + x_{j1} \qquad \forall \ j \geq 2, \tag{164}$$

$$z_{ij} = y_{ij} + x_{ij} \qquad \forall \ i, j \geq 2, \ i \neq j, \tag{165}$$

$$1 \leq u_j \leq (n-1) \qquad \forall \ j \geq 2, \tag{166}$$

$$x \text{ binary} . \tag{167}$$

The linear system defined by (155)—(167) can be further reduced by eliminating redundant constraints implied by others in the formulation. The constraint $z_{ij} \leq (n-1)x_{ij}$ in (160) is implied by $y_{ij} \leq (n-2)x_{ij}$ in (160) and by (165). The constraint $z_{ij} \geq 2x_{ij}$ in (159) is implied by $y_{ij} \geq x_{ij}$ in (159) and (165). The constraint $1 \leq u_j \leq (n-1)$, $j \geq 2$, in (166) is implied by (163) and (164). Lastly, the variable z_{ij} can be eliminated from the formulation using (165).

After eliminating the redundancies noted, and projecting the problem into the (x, u, y)-space by eliminating z_{ij} via (165), we arrive at the following system:

(RLT1) Minimize $\sum_i \sum_j c_{ij} x_{ij}$

subject to:

$$\sum_{j \neq i} x_{ij} = 1 \qquad \forall \ i = 1, \ldots, n, \tag{168}$$

$$\sum_{i \neq j} x_{ij} = 1 \qquad \forall \ j = 1, \ldots, n, \tag{169}$$

$$\sum_{\substack{j \neq i \\ j \neq 1}} y_{ij} + (n-1) x_{i1} = u_i \qquad \forall \ i \geq 2, \tag{170}$$

$$\sum_{\substack{i \neq j \\ i \neq 1}} y_{ij} + 1 = u_j \qquad \forall \ j \geq 2, \tag{171}$$

$$u_j \geq 1 + (n-3) x_{j1} + (1 - x_{1j}) \qquad \forall \ j \geq 2, \tag{172}$$

$$u_j \leq (n-2) - (n-3) x_{1j} + x_{j1} \qquad \forall \ j \geq 2, \tag{173}$$

$$x_{ij} \leq y_{ij} \leq (n-2) x_{ij} \qquad \forall \ i, j \geq 2, \ i \neq j, \tag{174}$$

$$y_{ij} + y_{ji} \geq u_j + (n-2) x_{ij} - (n-1)(1 - x_{ji}) \qquad \forall \ i, j \geq 2, \ i \neq j, \tag{175}$$

$$y_{ij} + y_{ji} \leq u_j - (1 - x_{ji}) \qquad \forall \ i, j \geq 2, \ i \neq j, \tag{176}$$

$$x \text{ binary} . \tag{177}$$

More importantly, we have the following.

Proposition 3. The re-formulation (RLT1) dominates Desrochers and Laporte's formulation (DL) in the sense that the constraints (123)–(127) are all implied by the linear system defined by (168)–(176).

Proof. The constraints given by (123), (124), (126) and (127) coincide with the constraints (168), (169), (172) and (173), respectively. To obtain (125), note that for any pair $i, j \geq 2$, $i \neq j$, examining (175) with i and j interchanged and multiplying by -1 yields upon adding the result to (176),

$$(y_{ij} - y_{ij}) + (y_{ji} - y_{ji}) \leq u_j - u_i - 1 + (3-n) x_{ji} + (n-1)(1 - x_{ij}) \qquad \forall \ i, j \geq 2, \ i \neq j.$$

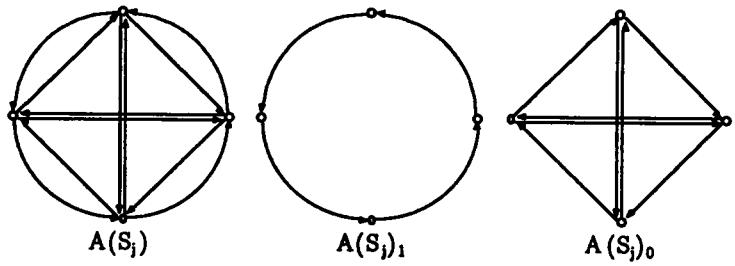

FIGURE 13. Arc sets for a subtour S_j.

This gives,

$$u_j \geq (u_i + 1) - (n-1)(1 - x_{ij}) + (n-3)x_{ji} \qquad \forall \quad i,j \geq 2,\ i \neq j,$$

which is (125). ■

Proposition 3 directly suggests that the **RLT1** approach described herein for the ATSP might also further tighten the lifted MTZ formulation for symmetric TSP cost matrices. Similar improvements are possible for the *vehicle routing problem* (VRP), the *distance constrained* VRP (DVRP), and the VRP *with time windows* (TWVRP) as examined by Desrochers and Laporte (1991) using the lifted MTZ framework.

4.3. Tightening DFJ via cycle inequalities

Apart from the MTZ subtour elimination constraints, the structure created by subtours themselves provides some useful special-structure sets that permit **RLT1** to achieve tighter representations. Suppose that we solve the underlying assignment problem on $D_n = (V, A)$, the complete digraph on n–vertices, given by (115), (116) and (119), and the solution forms subtours $S_j \subseteq V$, $2 \leq j \leq \lfloor \frac{n}{2} \rfloor$, $|S_j| \leq n-2$, for all j. Let $A(S_j) \subset A$ represent the set of all arcs connecting the nodes in subtour S_j. Denote by $A(S_j)_1 \subset A(S_j)$ those arcs in $A(S_j)$ for which $x_{ij} = 1$, $i,j \in S_j$, and $A(S_j)_0$ those arcs that currently have $x_{ij} = 0$, $i,j \in S_j$. Figure 13 illustrates these three sets for an arbitrary subtour on four vertices that are part of a complete digraph.

The special-structure set implied by each of these subtours, $S = S(j) \equiv \{x : \sum_{(i,k) \in A(S_j)} x_{ik} \leq |S_j| - 1 \ \forall\, j,\ x \in \{0,1\}\}$, provides a structure that can be effectively exploited by **RLT1**. When the arc set is restricted to only considering $A(S_j)_1$, then the inequalities of S are known as *cycle*

inequalities. Grötschel (1977) showed that all inequalities obtained by sequentially lifting the complementary arcs of each of these cycle inequalities in any order define facets of the convex hull of the monotone ATSP polytope $\tilde{P}_T^n$. It is tighter to initially consider the entire arc set $A(S_j)$ for inclusion in S. However, from a lifting initialization standpoint, similar to minimal covers for the knapsack polytope, it is advantageous to develop **RLT1** factors starting with the arc set $A(S_j)_1$, yielding the special-structure set $S = S(j) \equiv \{x : \sum_{(i,k)\in A(S_j)_1} x_{ik} \leq |S_j| - 1 \; \forall \; j, \; x \in \{0,1\}\}$. Theorem 4 establishes that the same facetial inequalities identified by Grötschel (1977) are obtainable via **RLT1**. For these cycle inequalities, **RLT1** yields tight constraints of the form

$$\sum_{(i,k)\in A(S_j)_1} x_{ik} + \sum_{(i,k)\in A(S_j)_0} \gamma_{ik} x_{ik} \leq |S_j| - 1. \tag{178}$$

The following example demonstrates the equivalence of the two techniques with regard to tightening cycle inequalities.

Example 1. Consider the cycle inequality for a three city subtour given by $x_{12}+x_{23}+x_{31} \leq 2$, in which the arc set of this subtour is given by $A(S_1)_1 = \{(1,2),(2,3),(3,1)\}$. A sequential lifting of the complementary asymmetric arcs currently not included in the subtour in the order $(2,1)$, $(1,3)$, and $(3,2)$ produces the tightened inequality $x_{12}+2x_{21}+x_{23}+x_{31} \leq 2$. Applying Theorem 4, the **RLT1** constraints via (107) are given by

$$\{x_{21} \cdot [2 - x_{12} - x_{23} - x_{31}]\}_L \geq 0 \qquad \text{and} \qquad \{(1 - x_{21}) \cdot [2 - x_{12} - x_{23} - x_{31}]\}_L \geq 0.$$

Using conditional logic on the first constraint under the condition $x_{21} \geq 1$, we obtain $x_{12} = x_{23} = x_{31} = 0$, yielding $x_{21}x_{12} = x_{21}x_{23} = x_{21}x_{31} = 0$, or

$$\{x_{21} \cdot [-x_{12} - x_{23} - x_{31}]\}_L \geq 0.$$

Summing this last inequality and the second **RLT1** constraint yields

$$2 - x_{12} - x_{23} - x_{31} - 2x_{21} \geq 0.$$

Continuing this process for each complementary arc $(i,k) \in A(S_j)_0$ yields exactly the same inequality produced via sequential lifting with $\gamma_{13} = \gamma_{32} = 0$. ■

The next proposition establishes the relationship between the various linear programming relaxations of the formulation (DFJ) along with MTZ subtour elimination constraints (120)–(122)

introduced thus far. Let $X_0 = X_{MTZ}$ represent the usual linear programming relaxation of (DFJ) with (117) replaced by (120)–(122), let X_{DL} represent the linear programming relaxation of (DL), and let X_{SD} represent the **RLT1** linear programming relaxation (RLT1). Then we have the following.

Proposition 4. $X_0 = X_{MTZ} \supseteq X_{DL} \supseteq X_{SD} \supseteq conv(X)$.

Proof. The relationship $X_{MTZ} \supseteq X_{DL}$ follows directly from the tightened formulation introduced by Desrochers and Laporte. By Remark 1, Desrochers and Laporte's reformulation is obtainable from a simple bound factor product type first level relaxation using **RLT0**. The dominance of **RLT1** over **RLT0** established by Sherali, et al. (1994) in Theorem 2 yields $X_{DL} \equiv X_{P_1} \supseteq \bar{X}_{P_1} \equiv X_{SD}$. Finally, $X_{SD} \supseteq \text{conv}(X)$ follows from the validity of (RLT1). ∎

4.4. Generating valid inequalities via quadratic terms

The formulation proposed by Desrochers and Laporte is essentially the result of employing a lifting technique that utilizes linear terms to achieve the desired tightening, while maintaining the validity of the newly generated constraints. In this section, we use *quadratic* terms to achieve a further tightening of non–facetial MTZ subtour elimination constraints. We demonstrate that it is also possible to apply this technique to facetial constraints in order to derive other strong valid inequalities.

The idea of using quadratic or multilinear terms to derive other strong valid inequalities, incorporates terms of the form $x_{ij}x_{jk}$ appended to existing facetial or non–facetial constraints along with appropriate constant coefficients that maintain feasibility with respect to the original problem. Contributions by such terms occur under more restrictive conditions than for similar linearly lifted terms appearing in the same constraints. That is, if a linear term x_{ij} appears in a constraint, then it is invoked by a feasible tour whenever this single arc connecting nodes i and j appears in this tour. However, for a quadratic term to have $x_{ij}x_{jk} = 1$, both arcs (i, j) and (j, k) must be present in a feasible tour so that $x_{ij} = 1$ and $x_{jk} = 1$. The presence of these quadratic terms corresponds to an exact description of a segment of a feasible tour passing through nodes i, j, and k in order (Figure 2).

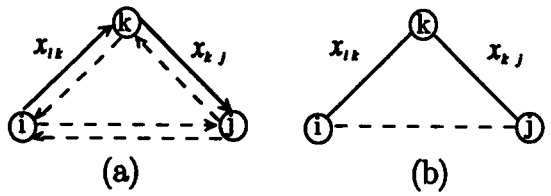

FIGURE 14. (a) Quadratic arcs present in the asymmetric TSP. (b) Quadratic arcs present in the symmetric TSP.

Proposition 5. The multilinear (quadratic) constraints

$$u_k \geq u_i + 2 - (n-1)(2 - x_{ij} - x_{jk}) + (n-2)(1 - x_{ij})(1 - x_{jk}) + (n-1)x_{ik} + (n-4)(x_{ki}x_{ij} + x_{jk}x_{ki} + x_{kj}x_{ji}) \tag{179}$$

for all distinct $i, j, k \geq 2$, are valid constraints for ATSP.

Proof. For any distinct triplet i, j, k, we can write the valid MTZ inequality (79) corresponding to $i \to j$, and $j \to k$ as $u_j \geq u_i + 1 - (n-1)(1-x_{ij})$ and $u_k \geq u_j + 1 - (n-1)(1-x_{jk})$, respectively. Surrogating these two constraints yields the valid inequality

$$u_k \geq u_i + 2 - (n-1)(2 - x_{ij} - x_{jk}). \tag{180}$$

We will now show that (179) can be obtained by lifting (180) in a term by term fashion. To this end, consider first a tightening of (180) by finding the maximum value of $\theta \geq 0$ such that

$$u_k \geq u_i + 2 - (n-1)(2 - x_{ij} - x_{jk}) + \theta(1 - x_{ij})(1 - x_{jk})$$

is valid when $x_{ij} = x_{jk} = 0$. Such a value of θ is given by solving the problem

$$\theta = \min\{u_k - u_i - 2 + (n-1)(2 - x_{ij} - x_{jk}) : 1 \leq u_k \leq n-1,$$
$$1 \leq u_i \leq n-1,\ x_{ij} = x_{jk} = 0,\ x \text{ binary }\}.$$

This yields the value

$$\theta = 1 - (n-1) - 2 + (n-1)2 = (n-2).$$

For the coefficient of x_{ik}, notice that when $x_{ik} = 1$, $u_k = u_i + 1$, and $x_{ij} = x_{jk} = 0$, so that we have

$$\theta = \min\{u_k - u_i - 2 + (n-1)2 - (n-2)\} = (n-1).$$

For each of the remaining terms, $x_{ki}x_{ij}$, $x_{jk}x_{ki}$, and $x_{kj}x_{ji}$, we get the following. For $x_{ki} = x_{ij} = 1$, we have $u_i = u_k + 1$, $x_{jk} = 0$, $x_{ik} = 0$, and the other two quadratic terms are zeroes. This gives the valid coefficient

$$\theta \leq -1 - 2 + (n-1) = (n-4).$$

For $x_{jk} = x_{ki} = 1$, we get $u_i = u_k + 1$, $x_{ij} = 0$, $x_{ik} = 0$, and the other two quadratic terms $x_{ki}x_{ij}$ and $x_{kj}x_{ji}$ are zeroes. Hence, we again obtain

$$\theta \leq -1 - 2 + (n-1) = (n-4).$$

Similarly, for $x_{kj} = x_{ji} = 1$, we get $u_i = u_k + 2$, $x_{ij} = 0$, $x_{jk} = 0$, $x_{ik} = 0$, and the other two quadratic terms $x_{ki}x_{ij}$ and $x_{jk}x_{ki}$ are zeroes. This yields

$$\theta \leq u_k - u_i - 2 + n = (n-4).$$

Since the value of θ can, in each of the cases noted, be taken as equal to $(n-4)$ for each of the quadratic terms, this establishes the validity of (179), and concludes the proof. ■

Directly incorporating quadratic terms into a TSP formulation increases the complexity of the problem, and is therefore not recommended in practice. However, notice that we can replace each quadratic term in the constraints of Proposition 5 by a linear expression. For example, $x_{ki}x_{ij}$ can be replaced by $(x_{ki} + x_{ij} - 1)$ since $x_{ki}x_{ij} \geq x_{ki} + x_{ij} - 1$ while maintaining validity.

4.5. Tightening the 3-index TSP

This section introduces a new, RLT generated, 3-index formulation of TSP based on a quadratic assignment problem (QAP) formulation of TSP. This new formulation contains valid inequalities not available in any of the current 3-index formulations, and results in a tightened linear programming relaxation.

One 3-index traveling salesman problem formulation, known as the *time dependent* TSP, was originally proposed as a formulation for the 1-machine, n-job, scheduling problem and was studied by Fox (1973) in his dissertation. Fox, Gavish, and Graves (1980) later presented a formulation of this time dependent TSP as follows. For each time period $t = 1, \ldots, n$, define c_{ijt} as the cost of the salesman traveling from city $i \rightarrow j$ in period t. Let $x_{ijt} = 1$ if the salesman travels from city

i to city j in time period t, and $x_{ijt} = 0$ otherwise. Note that it is assumed that every feasible tour starts in city 1 in time period $t = 1$.

(TD) $$\text{Minimize} \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{t=1}^{n} c_{ijt}x_{ijt}$$

subject to:

(181) $$\sum_{j=1}^{n}\sum_{t=1}^{n} x_{ijt} = 1 \qquad \forall \quad i = 1, \ldots, n,$$

(182) $$\sum_{i=1}^{n}\sum_{t=1}^{n} x_{ijt} = 1 \qquad \forall \quad j = 1, \ldots, n,$$

(183) $$\sum_{i=1}^{n}\sum_{j=1}^{n} x_{ijt} = 1 \qquad \forall \quad t = 1, \ldots, n,$$

(184) $$\sum_{j=1}^{n}\sum_{t=2}^{n} t x_{ijt} = \sum_{j=1}^{n}\sum_{t=1}^{n} t x_{jit} + 1 \qquad \forall \quad i = 2, \ldots, n,$$

(185) $$x \text{ binary}.$$

The constraints (184) insure that the time period in which city i is entered by the salesman is one less than the time period it is exited. Fox, et al. (1980) further demonstrated two additional characteristics that provide an even more compact formulation. First, they note that constraints (183) are superfluous to the formulation, a fact that we further validate in the process of deriving a tightened formulation directly from the quadratic assignment problem formulation. Second, they note that (181)–(183) can equivalently be replaced by

(186) $$\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{t=1}^{n} x_{ijt} = n,$$

reducing the number of constraints from $4n - 1$ to n. However, as stated in the introduction, integer programming problems such as TSP benefit by the addition of constraints that tighten the linear programming relaxation. Thus, compactness is not necessarily a desirable trait, unless the reduced set of constraints possess desirable characteristics from the viewpoint of retaining a tight formulation and preserving some special structure that can be exploited.

If we assume that the cost of traveling between cities is not time dependent, and let d_{jl} represent the cost of the salesman traveling from city $j \rightarrow l$ regardless of time period, we arrive at a second

well known *standard* 3-index formulation of the TSP. Define $y_{ijl} = 1$ if the salesman travels from city j to city l on the i^{th} leg of the tour, and $y_{ijl} = 0$ otherwise. Then, this standard formulation can be stated as follows.

(3ID) Minimize $\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{l\neq j} d_{jl}y_{ijl}$

subject to:

(187) $$\sum_{j=1}^{n}\sum_{l\neq j} y_{ijl} = 1 \qquad \forall \ \ i = 1,\ldots,n,$$

(188) $$\sum_{i=1}^{n}\sum_{l\neq j} y_{ijl} = 1 \qquad \forall \ \ j = 1,\ldots,n,$$

(189) $$\sum_{i=1}^{n}\sum_{j\neq l} y_{ijl} = 1 \qquad \forall \ \ l = 1,\ldots,n,$$

(190) $$\sum_{j\neq l} y_{ijl} = \sum_{j\neq l} y_{i+1,l,j} \qquad \forall \ \ i, l,$$

(191) y binary .

Again, we can show that any one of the constraint sets (187), (188), or (189) is implied in the continuous sense by the remaining constraints, and hence, can be deleted.

The constraints (190) represent subtour elimination constraints for this problem formulation. To demonstrate their effectiveness, consider the case of $n = 5$ in which two subtours exist: $ST_1 = \{1,2\}$ and $ST_2 = \{3,4,5\}$. When $i = 2$ and $l = 1$, the left-hand side of (190) has $\sum_{j\neq 1} y_{2j1} = 1$, since $y_{221} = 1$. However, for $i = 3$ with $l = 1$, the right-hand side equals zero since there are no arcs leaving node $l = 1$ on the third leg of the tour, thus violating (190) and defeating the subtour. For any feasible Hamiltonian circuit, however, constraints (190) are satisfied.

For the purposes of introducing the new 3-index formulation, note that a related formulation is available in the form of a quadratic assignment problem (QAP) by letting $x_{ij} = 1$ if the ith

city visited by the salesman is city j for all i, j, and $x_{ij} = 0$ otherwise:

(QAP) Minimize $\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{l\neq j} d_{jl}x_{ij}x_{i+1,l}$

subject to:

(192) $$\sum_{j=1}^{n} x_{ij} = 1 \qquad \forall \quad i = 1, \ldots, n,$$

(193) $$\sum_{i=1}^{n} x_{ij} = 1 \qquad \forall \quad j = 1, \ldots, n,$$

(194) y binary .

In light of the linearization concepts previously presented, the relationship between (3ID) and (QAP) is apparent, and serves as the starting point in deriving the new 3-index formulation. Hence, suppose we apply the standard RLT linearization substitution $y_{ijl} = x_{ij}x_{i+1,l}$ for all $i, j, l \neq j$, recognizing that this substitution is accomplished modulo n over the index set i. As usual, we have the additional linearization constraints associated with this substitution: $y_{ijl} \geq 0$, $y_{ijl} \leq x_{ij}$, $y_{ijl} \leq x_{i+1,l}$, and $y_{ijl} \geq x_{ij} + x_{i+1,l} - 1$. Additionally, we can apply RLT level one products, being careful to create only products of the form $x_{ij}x_{i+1,l}$ present in the QAP. Accordingly, we form the following products using constraint (192):

$$\left(1 - \sum_{j} x_{ij}\right) \cdot x_{i+1,k} \geq 0 \qquad \forall\, i, k$$

$$\left(1 - \sum_{j} x_{ij}\right) \cdot x_{i-1,k} \geq 0 \qquad \forall\, i, k,$$

yielding the respective linearized equality constraints

(195) $$\sum_{j\neq k} y_{ijk} = x_{i+1,k} \qquad \forall\, i, k$$

(196) $$\sum_{j\neq k} y_{i-1,kj} = x_{i-1,k} \qquad \forall\, i, k.$$

Notice that (195) and (196) imply the linearization constraints $y_{ijl} \leq x_{i+1,l}$ and $y_{ijl} \leq x_{ij}$.

Therefore, we obtain the intermediate formulation

(QAP2) Minimize $\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{l\neq j} d_{jl}y_{ijl}$

subject to:

(197) $$\sum_{j=1}^{n} x_{ij} = 1 \qquad \forall\ i,$$

(198) $$\sum_{i=1}^{n} x_{ij} = 1 \qquad \forall\ j,$$

(199) $$\sum_{j\neq k} y_{ijk} = x_{i+1,k} \qquad \forall\ i,k,$$

(200) $$\sum_{j\neq k} y_{i-1,kj} = x_{i-1,k} \qquad \forall\ i,k,$$

(201) $$y_{ijl} \geq x_{ij} + x_{i+1,l} - 1 \qquad \forall\ i,j,l\neq j,$$

(202) $$y \geq 0, \qquad x \text{ binary}.$$

Although this is primarily an intermediate formulation for the purposes of this section, it is worthwhile to note that (QAP2) provides an assignment structure via the x-variables, and permits the y-variables to be treated as continuous variables. This can be convenient in branching and cutting plane generation strategies such as implicit Benders' cuts via reduced costs of linear programming relaxations as proposed by Johnson and Suhl (1980).

As a result of this intermediate formulation, it is possible to project out the x variables using the relationships established in the constraint set. From (200), we obtain $x_{ik} = \sum_{j\neq k} y_{ikj}$, or

(203) $$x_{ij} = \sum_{l\neq j} y_{ijl} \qquad \forall\, i,j,$$

which enables a restatement of (200) as

(204) $$x_{i+1,l} = \sum_{j\neq l} y_{i+1,lj} \qquad \forall\, i,l.$$

Similarly, (199) gives

(205) $$x_{i+1,l} = \sum_{j\neq l} y_{ilj} \qquad \forall\, i,l.$$

Substituting the expression for x_{ij} from (203) into (197) and (198), we obtain the two expressions

$$\sum_j \sum_{l \neq j} y_{ijl} = 1 \quad \forall i \qquad \text{and} \qquad \sum_i \sum_{l \neq j} y_{ijl} = 1 \quad \forall j. \tag{206}$$

When y_{ijl} assumes binary values, (206) enforces that x_{ij}, defined by (203), will be binary as well, while satisfying the assignment constraints (197) and (198). Thus,

$$\sum_{j \neq l} y_{ijl} = \sum_{j \neq l} y_{i+1,jl} \qquad \forall i, l. \tag{207}$$

Combining (206) and (207), and substituting (203) and (205) into constraint (201), we arrive at the following new 3-index TSP formulation:

(RLT3ID) Minimize $\sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{l \neq j} d_{jl} y_{ijl}$

subject to:

$$\sum_{j=1}^{n} \sum_{l \neq j} y_{ijl} = 1 \qquad \forall \ i, \tag{208}$$

$$\sum_{i=1}^{n} \sum_{l \neq j} y_{ijl} = 1 \qquad \forall \ j, \tag{209}$$

$$\sum_{j \neq l} y_{ijl} = \sum_{j \neq l} y_{i+1,l,j} \qquad \forall \ i, l, \tag{210}$$

$$y_{ijl} \geq \sum_{t \neq j} y_{ijt} + \sum_{m \neq l} y_{iml} - 1 \qquad \forall \ i, j, l \neq j, \tag{211}$$

$$y \text{ binary.} \tag{212}$$

Notice the correspondence between (RLT3ID) and (3ID). Observe that (188) are superfluous, being implied by (189) and (190), as seen simply by summing (190) over the index i and using (189). (This relationship was exploited implicitly by Fox, et al. (1980), who left (188) out of the construction of their aggregate constraint (186).) However, (211) appears to be a new addition to the standard formulation, available via specific RLT products of the type presented. The strength of this new formulation for providing tight upper bounds for asymmetric TSP is compared with those available via the linear programming relaxation of the standard formulation in the next chapter. We also note, although do not include in this study, that a complete first level **RLT0** or **RLT1** application to QAP(TSP) is possible, in which all products $x_{ij}x_{pq}$ are created via

multiplications using (192) and (193). Such a complete application may reveal other structures not yet identified. Lastly, if the intermediate formulation (QAP2) is used, then the relationship between the binary x variables and the continuous y variables via (203) should be recognized and exploited. Enforcing x binary and preserving feasible completion to the assignment constraints within a branch-and-bound or other enumeration scheme has the effect of setting entire groups of y variables to either zero or one via (203), using (197) and (198).

CHAPTER 5

Computational Results

In this chapter, we present the methodology and results for two computational experiments relevant to the theoretical material presented in earlier chapters. It is important to note that the computational experiments herein were designed with the purpose of exploring the effectiveness of **RLT1** in closing an existing LP–IP gap for various structural configurations of the set packing problem with clique constraints and the five different formulations of the aysymmetric traveling salesman problem (ATSP) introduced previously. In this sense, the results reflect the performance of **RLT1** using the specific software and system configurations stated, and are not intended to showcase an implementation of the **RLT1** methodology, which is under development as of this writing. An actual implementation of **RLT1** would be designed to specifically exploit the type of structure generated by an **RLT1** reformulation, as opposed to simply using the techniques afforded by the commercial software used.

In reporting on both experiments, we have attempted to adhere to the guidelines on reporting computational experiments proposed by Crowder, Dembo and Mulvey (1979), and Greenberg (1990) as closely as possible. Since this study is intented to be a stepping–stone for future research, as much of the important detail is provided so that the techniques introduced in this study can be reproduced within the specified parameters. Additionally, it is our intent to extend this reformulation-linearization strategy to a much larger class of test problems and computing environments in follow–on research. Therefore, it is equally as important to highlight any shortcomings of **RLT1** as evident from the testing as it is to showcase exceptional performance.

The first experiment examines the relative performance of a level–one application of both **RLT1**

and **RLT0** versus the usual linear programming relaxation for various set packing problems. Using a set of pseudo-randomly generated set packing problems of the type

$$\text{Max}\{\sum_{j=1}^{n} c_j x_j \ : \ \sum_{j \in N_i} x_j \leq 1 \quad \forall\, i = 1, \ldots, m,\ x \text{ binary}\},$$

where $0 < c_{ij} \leq 100$, $c_{ij} \in Z^+$ for all i, j, we examine the effectiveness of both **RLT0** and **RLT1** in closing a known LP-IP gap existing between the upper bound obtained from the LP relaxation and the proven optimal integer solution for different problem instances. Note that these set packing problems could also be viewed as clique-cover representations of some underlying vertex packing problems defined with respect to their corresponding intersection graphs. The results obtained can therefore be interpreted as arising in the context of solving vertex packing problems as well.

The second experiment consists of two separate sets of computational tests. The first set of experiments compares the lower bounds achieved by solving the LP relaxations of the five formulations of the ATSP introduced in Chapter 4: the standard TSP formulation with MTZ subtour elimination constraints, the formulation of Desrochers and Laporte (1991), the **RLT1** tightened MTZ formulation (RLT1), the standard 3–index TSP formulation (3ID), and the **RLT1** tightened 3–index formulation (RLT3ID). The goal of this experiment is to assess the relative performance of the new **RLT1** reformulations in comparison with previous work. Randomly generated asymmetric cost matrices of various sizes were used to create each problem instance. Again, the problems were selected for inclusion in the test set based on possessing a sizable LP–IP gap. The second experiment compares the lower bounds achieved by the standard TSP formulation with MTZ subtour elimination constraints, the formulation of Desrochers and Laporte (1991), and the **RLT1** tightened MTZ formulation (RLT1), when applied to three standard test problems obtained from the TSPLIB at Rice University. The goal of this experiment is to examine the performance of these formulations when applied to reportedly difficult problem instances. These test problems represent three of the current six instances of ATSP included in the TSPLIB, and were made available by M. Fischetti and B. Repetto.

All computations were performed on an IBM RS 6000 computer, running the AIX version 3 operating system. The General Algebraic Modeling System (GAMS), version 2.25 (1992), was used to generate the models. The IBM Optimization Subroutine Library (OSL) MIP solver,

version 2.001, was used to solve the 0-1 set packing problems, the ATSP problems, as well as their various relaxations.

5.0.1. Experimental design. The set packing test problems were carefully constructed using a uniform distribution pseudo-random number generator to obtain a specific level of density for the constraint coefficient matrix. Sherali and Adams (1986) hypothesized that the density of the constraint set may affect the strength of the linearization of **RLT0**, becoming generally stronger with a lower density. This would imply that, in applying either **RLT0** or the special-structures hierarchy of **RLT1**, a judicious choice among all available factors should be made to create an efficient and effective reformulation. As a result, as explained in the section on implementation, the density of each test problem instance was controlled in such a manner that once a nonzero appeared for any particular element of the constraint matrix, it did not change as the density parameter increased. As opposed to the more general hypothesis of Sherali and Adams (1986), we were specifically interested in assessing how the *connectivity* of the coefficient matrix, via the density level, affected the performance of both **RLT1** and **RLT0**. Connectivity is a measure of the linkage between constraints, caused by variables appearing in common, and is recognizably dependent upon the pattern of nonzero elements. By controlling density in the manner that we did, we could guarantee that connectivity would be monotonically non-decreasing as the level of density increased. For reasons elaborated in the sequel, when the density of the constraint matrix approaches its extremes at 100% and 0%, **RLT1** and **RLT0** have no effect on tightening the linear programming relaxation.

The dimensionality of the set packing coefficient matrices were varied in both size ($m \times n$) and shape (rectangular orientation or square). The test problems used possess coefficient matrices with either $m < n$, or $m = n$. This structure is decidedly more representative of the structure of set packing problems in general, and encompasses the structures seen in the related maximum clique constraints along with generalized upper bounding constraints.

The random test problems used to assess the performance of the five formulations of ATSP presented in Chapter 4 were generated in a manner similar to those used in the set packing problems. An $(n \times n)$ cost, or distance matrix, C, associated with a complete graph on n vertices, was created using a uniform distribution pseudo-random number generator on $U(1, 100)$. We

were specifically interested in assessing the performance of each formulation for different sized problem instances, each of which possessed a significant LP–IP gap. The sizes used for these computations were chosen to be large enough to challenge the automatic reformulation process and the gap closing abilities of each formulation, yet small enough to prevent exceeding the memory limitations of the RS 6000.

The second set of test problems were taken from the library TSPLIB at Rice University. This is an electronic library of publicly available traveling salesman and related problem instances accessible via INTERNET. Each of these test problems were available in full matrix explicit edge-weight format, along with their known integer optimal solutions. We use these problem instances to compare the lower bounds attained by the standard TSP formulation with MTZ subtour elimination constraints, the formulation of Desrochers and Laporte (1991), and the **RLT1** tightened MTZ formulation (RLT1). Both the standard 3-index and the **RLT1** tightened ATSP formulations were not used for this experiment, because the data structures generated as a result of the triple indexed summations exceeded the memory capabilities of the RS 6000. As noted by Adams and Sherali (1993), special Lagrangian relaxation strategies need to be used with such large-sized reformulations.

5.0.2. Measures of performance. For each set packing problem instance, we computed the optimal value of the 0-1 packing problem (IP), that of its ordinary LP relaxation, as well as the optimal values of the first-level relaxations produced by applying **RLT0** and **RLT1**, where the latter was generated by using all of the defining clique constraints to represent the set S. Additionally, we measured the number of simplex iterations required for each problem to achieve optimality, and for one set of computations, a count of the number of variables assuming fractional values was taken.

For each ATSP problem instance, we computed the optimal value each formulation's LP relaxation, as well as the number of simplex iterations required for each problem to achieve optimality.

5.0.3. Implementation. The computational process consisted of the three phases shown in Figure 15: model specification, model generation, and model solution. During the model specification phase, an input file sets the value of several important parameters that control both

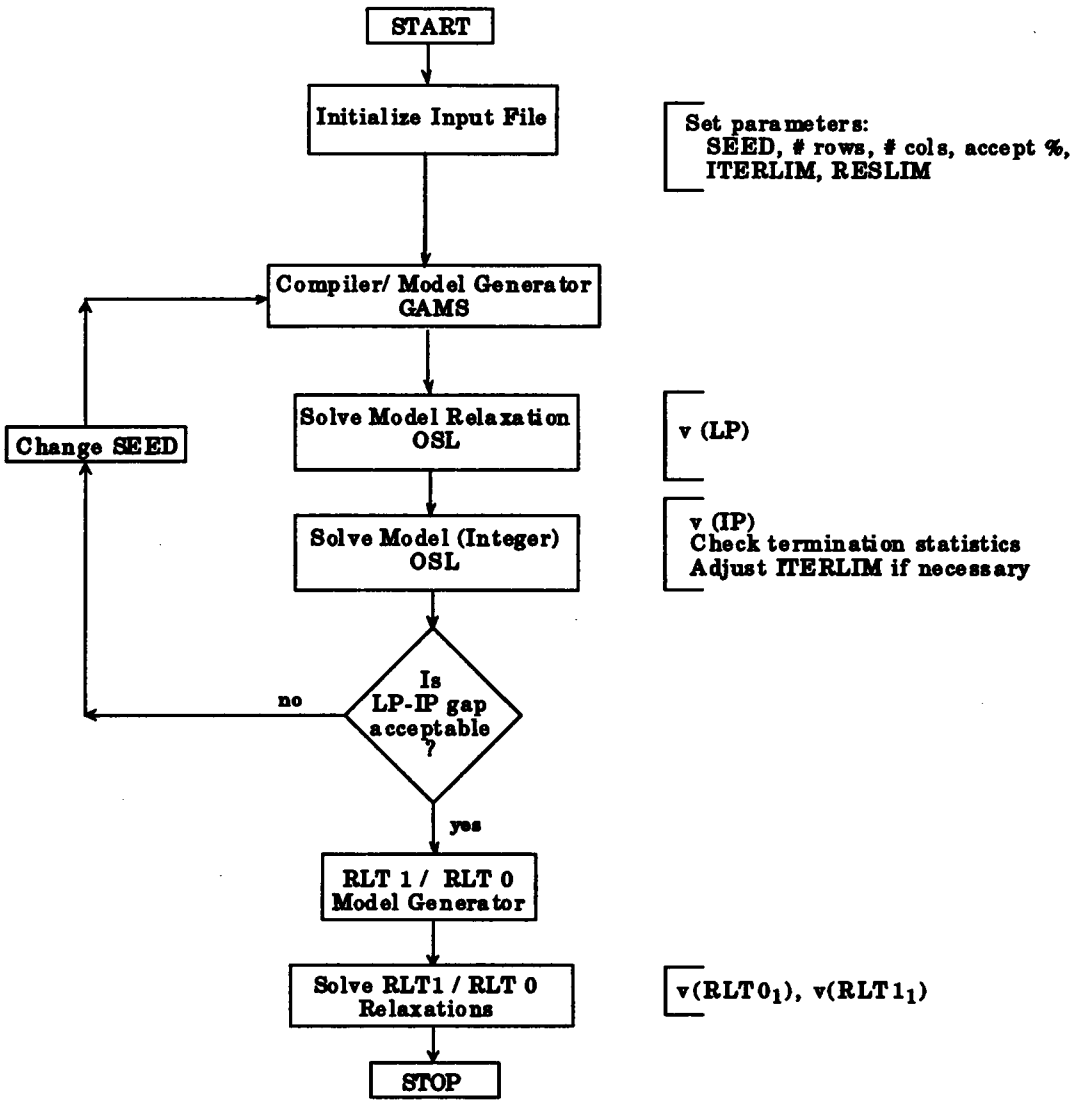

FIGURE 15. Set packing problem methodology.

the model being generated and the solution procedure used:

- Problem size. The number of rows (m) and columns (n) of the set packing constraint matrix to be generated, and the value of n for the ATSP cost matrix.
- Density parameter (accept). A scalar threshold value called 'accept' is used in the set packing test problems to test if a component of the randomly generated constraint matrix should be nonzero or not. This parameter is not used for the ATSP test problems.
- Iteration limit (ITERLIM). This number restricts the number of iterations that can be executed by either the simplex method or branch-and-bound before the solver will terminate its efforts. For all problems, ITERLIM= 100000 iterations.
- Optimality criteria (OPTCR). The relative optimality criteria used by the OSL MIP solver to obtain $v(IP)$ used in the calculation of '% gap.' The solution procedure stops if it can guarantee that the best solution is within (100× OPTCR) % of the global optimal. For all problems, OPTCR= 0.0, so that a provenly optimal solution was obtained.
- Seed (SEED). The value of the pseudo-random number generator seed was changed for each problem instance so that a larger sized problem was not simply an expanded version of an earlier instance.

Since RLT is an automatic reformulation technique, the input file for the set packing test problems identifies which sets of inequalities are reformulated and included in any particular run. This enables the specification of **RLT1** and **RLT0** constraints to be included in the same input file. Different models are created by simply switching the appropriate constraint sets to be 'on' or 'off.' For the ATSP test problems, the specific reformulations have already been completely performed in Chapter 4, and were directly coded into their input files. The general input file code for each class of problems are included in the appendix for reference.

In the second phase, the model specification is passed to the GAMS compiler for syntax checking and model generation. In the GAMS environment, this phase serves as a link between the model specifications set by the declarations and definitions of the input file, and the data structures required by the OSL solver. During this phase, a uniform pseudo-random number generator is used to create the cost coefficients c_j, $j = 1, \ldots, n$, and the constraint coefficient matrix A for the set packing test problems. The cost coefficients are selected from a uniform distribution

Ordinary LP relaxation				Level-one of RLT0			Level-one of RLT1		
(m,n)	DENSITY	% gap	ITER	(m',n')	% gap	ITER	(m',n')	% gap	ITER
(15, 25)	66%	42.0	22	(170, 25)	16.8	160	(123, 25)	0	35
(20, 30)	56%	26.7	36	(319, 30)	7.2	267	(214, 30)	0	32
(25, 35)	52%	57.3	43	(481, 35)	20.2	152	(328, 35)	0	113
(55, 45)	27%	37.0	103	(2256, 63)	2.0	344	(1896, 63)	0	171
(35, 35)	56%	36.5	48	(631, 35)	10.7	292	(607, 35)	0	60

TABLE 6. Computational results for set packing problems.

$U(1,100)$. To insure monotonicity of connectivity, the elements of A are first selected from a uniform distribution $U(0,1)$. Then, each a_{ij}, $i = 1,\ldots,m$, $j = 1,\ldots,n$, is compared with the preset value of accept specified in the input file. If $a_{ij} <$ accept, the value of a_{ij} is set equal to 1, and set equal to zero otherwise. In this manner, the constraint matrix can be set at a particular density level of (100× accept)%. For the ATSP test problems, this phase creates the cost or distance matrix where each c_{ij} is uniformly distributed on $U(1,100)$. In the final phase, each model's data structure is passed to the OSL solver to obtain the required optimal solution.

5.0.4. Results for set packing problems. Table 6 gives the percentage gaps between the upper bounds obtained from the ordinary LP relaxation, a level-one **RLT0** application, and a level-one **RLT1** application, with respect to the optimal 0-1 value, along with the sizes of the respective relaxations (m' = number of constraints, n' = number of variables), and the number of simplex iterations (ITER) needed by the OSL solver to achieve optimality. It is interesting to note that in all instances examined, *the first level* **RLT1** *was sufficient to solve the integer program.* In fact, this was the case for all problems examined, independent of inclusion within the test set reported in Table 6. On the other hand, although **RLT0** appreciably improved the upper bound produced by the ordinary LP relaxation, it still left a significant gap that remains to be resolved in these problem instances. Moreover, except for a small number of problems, the relatively simpler structure of **RLT1** resulted in far fewer simplex iterations being required to solve this relaxation as compared with the effort required to solve **RLT0**.

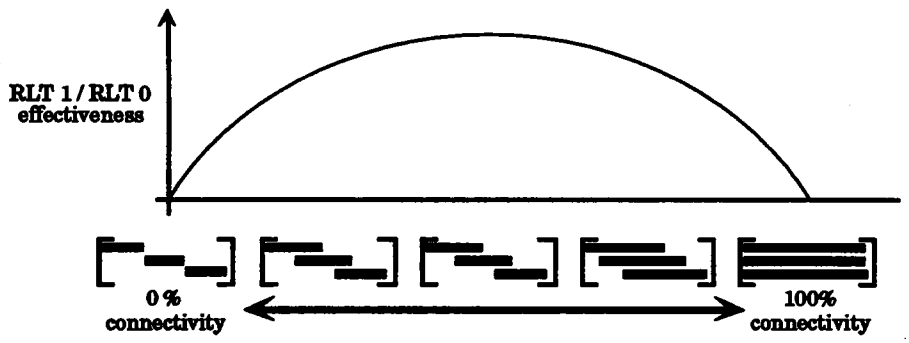

FIGURE 16. Effectiveness dependency on connectivity.

5.0.5. Connectivity and RLT. In generating random density matrices, the issues as to whether or not **RLT0** matches the performance of **RLT1** in closing the LP–IP gap appeared to be dependent upon the degree of connectivity of the constraints, as reflected by the particular density of nonzeros in each original constraint matrix. We therefore conjectured that the difference in effectiveness in closing the LP–IP gap between first level applications of **RLT1** and **RLT0** should vary in a manner similar to the spectrum illustrated in Figure 16, for the following reasons.

When no connectivity exists within a set of set packing constraints, they are completely separable. Hence, the set packing problem decomposes into p knapsack problems.

At the opposite end of the spectrum, where the constraints possess total connectivity, the constraint structure has the property that $\bigcap_{i=1}^{p} N_i = N$, *i.e.*, all constraints are identical. This set of redundant constraints reduces to a single knapsack constraint $\sum_{j \in N} x_j \leq 1$ containing all variables, in which no RLT multiplier has any useful application. Therefore, neither **RLT1** nor **RLT0** induces any tightening effect in either extreme situation. These effects are confirmed by the computational results seen in Table 7 and Table 8.

Throughout the intermediate range between the two extremes, connectivity and density have a variable affect on **RLT1** and **RLT0** that depends upon whether or not the original x-variables are fractionating in the ordinary LP relaxation optimal solution. As an example, consider the case of the single constraint, $x_1 + x_2 + x_3 \leq 1$, being multiplied by both **RLT0** and **RLT1** multipliers derived from a second constraint, $x_4 + x_5 + x_6 \leq 1$. For **RLT0**, each valid multiplier at level one

produces the constraints

$$(1 - x_1 - x_2 - x_3) \cdot (x_j) \geq 0,$$

$$\text{and} \quad (1 - x_1 - x_2 - x_3) \cdot (1 - x_j) \geq 0 \qquad j = 4, 5, 6.$$

This multiplication results in the constraints

$$\alpha \equiv x_j - w_{1j} - w_{2j} - w_{3j} \geq 0 \tag{213}$$

$$\text{and} \quad (1 - x_1 - x_2 - x_3 - \underbrace{[x_j - w_{1j} - w_{2j} - w_{3j}]}_{\alpha \geq 0}) \geq 0, \qquad j = 4, 5, 6. \tag{214}$$

The nonnegativity of (213) is more difficult to achieve due to the presence of w-variables. Thus, the nonnegative quantity α being subtracted from (214) tightens this constraint. In comparison, a level one application of **RLT1** with $\{x_4 + x_5 + x_6 \leq 1,\ x_j \geq 0,\ j = 4, 5, 6\}$ being a part of S, generates the constraints

$$x_j - w_{1j} - w_{2j} - w_{3j} \geq 0, \tag{215}$$

and

$$1 - x_1 - x_2 - x_3 - \left[\sum_{j=4}^{6} (x_j - w_{1j} - w_{2j} - w_{3j})\right] \geq 0 \qquad j = 4, 5, 6. \tag{216}$$

The difficulty of achieving nonnegativity is increased for these new **RLT1** constraints. Note however, that the benefits of this type of constraint tightening are only realized when the x_j, $j = 4, 5, 6$, are at fractional values in the LP solution. When the x_j, $j = 4, 5, 6$, assume binary values, the effects of tightening due to the presence of these nonnegative terms are lost. When $x_j = 1$, $j = 4, 5, 6$, both sides of (213) and (215) are zero. However, when $0 < x_j < 1$, (213) and (215) are tightened as a result of the forced nonnegativity of the RLT terms.

Since this effect is more pronounced in the **RLT1** constraints than those generated by **RLT0**, we would expect the **RLT1** formulation to be more effective in practice than **RLT0** in closing the LP–IP gap. Thus, in varying the density of a particular set packing problem, we would expect **RLT0** to perform as well as **RLT1** whenever the effectiveness of the latter is low due to extreme values of density, and we would expect **RLT1** to dominate **RLT0** in the middle ranges. Again, this effect is witnessed in Table 7 and Table 8.

(15 × 25)	Standard LP		Level-one RLT0		Level-one RLT1	
density (%)	% gap	ITER	% gap	ITER	% gap	ITER
5	0	28	0	355	0	176
10	0	23	0	305	0	228
15	0.8	32	0	487	0	418
20	0	25	0	313	0	354
25	0	20	0	359	0	283
30	13.9	23	0	283	0	268
35	39.5	22	11.4	226	0	228
40	17.7	20	0	116	0	57
45	6.1	28	0	36	0	36
50	0	20	0	31	0	42
55	50.2	16	18.1	67	0	53
60	42.9	22	16.8	160	0	35
65	41.7	29	7.7	80	0	17
70	23.2	22	0.1	32	0	19
75	6.6	31	0	56	0	23
80	6.6	41	0	21	0	18
85	0	13	0	21	0	15
90	0	20	0	20	0	7
95	0	8	0	8	0	9

TABLE 7. Density response spectrum.

(35 × 35)	Standard LP		Level-one RLT0		Level-one RLT1	
density (%)	% gap	ITER	% gap	ITER	% gap	ITER
5	0	38	0	478	0	511
10	0	38	0	1434	0	910
15	4.9	63	0	2075	0	2498
20	10.5	68	0	1121	0	1144
25	15.8	62	0	816	0	796
30	9.5	57	0	325	0	146
35	48.0	60	13.0	437	0	177
40	67.5	60	31.8	312	0	152
45	56.9	52	25.7	291	0	159
50	49.6	23	18.6	271	0	68
55	36.5	48	10.7	292	0	60
60	17.7	35	0	135	0	70
65	9.6	36	0	122	0	35
70	3.1	38	0	115	0	21
75	0	31	0	31	0	18
80	0	53	0	47	0	15
85	0	33	0	20	0	14
90	0	23	0	16	0	12
95	0	11	0	11	0	11

TABLE 8. Density response spectrum.

An unexpected result occurs when *no* w-variables survive cancellation due to adjacency, and the density level is away from the extremes. The difference in the gap closing abilities of **RLT1** and **RLT0** becomes acute, and **RLT1** strictly dominates **RLT0**. This effect can be explained by the difference in structure between the **RLT1** clique factors and **RLT0** bound factor complements, and the amount of *adjacency* occurring in each constraint. The concept of 'adjacency' is defined in the following manner. Let the variables x_j and x_k represent distinct vertices in the set packing graph. Then, x_j and x_k are said to be adjacent if both $j, k \in N_i$ for some $i = 1, \ldots, m$. That is to say, both x_j and x_k appear in the same constraint somewhere in the system of inequalities. Cancellation of their associated higher order RLT w-variables occurs because both variables cannot assume the value 1 simultaneously in the set packing constraints. When no w-variables survive cancellation, the resulting higher dimensional constraint contains the original x-variables, augmented by those x-variables present in the RLT multiplicative factors. For **RLT0**, this produces constraints of the form $1 - \sum_{j \in N_i} x_j - x_k \geq 0$, as a result of multiplication by the bound factor complement $(1 - x_k) \geq 0$, $k \notin N_i$. An **RLT1** clique factor inherits at least as many new x-variables as does **RLT0**, simply because of the number of variables present in the clique multipliers. For **RLT1**, this yields constraints of the form $1 - \sum_{j \in N_i} x_j - \sum_{j \in N_k - N_i} x_j \geq 0$, $N_i \neq N_k$. This effect is significant because the resulting RLT constraint is tightened in the space of original variables, *independent of any projection operation.*

For example, consider the set of three constraints given by

$$x_1 + x_2 + x_4 \leq 1$$

$$x_3 + x_4 + x_5 \leq 1$$

$$x_1 + x_2 + x_3 + x_5 \leq 1.$$

In a complete level-one application of **RLT1**, multiplying the first constraint by the **RLT1** clique factor $(1 - x_3 - x_4 - x_5) \geq 0$ derived from the second constraint, yields

$$x_1 + x_2 + x_3 + x_4 + x_5 \leq 1,$$

since $w_{13} = w_{23} = w_{34} = w_{15} = w_{25} = w_{45} = w_{14} = w_{24} = 0$ under the assumed binariness of the variables used in **RLT1** factors and the relationships present in the remaining clique

(15 × 25)	LP	RLT0		
density (%)	x-frac	w-frac	x-frac	conv(X)
15	17	0	0	yes
20	7	0	0	yes
25	9	0	0	yes
30	6	0	0	yes
35	6	18	15	no
40	11	0	17	no
45	10	0	16	no
50	12	0	15	no
55	5	0	11	no
60	6	0	11	no
65	4	0	13	no
70	3	0	8	no
75	5	0	0	yes
80	5	0	0	yes

TABLE 9. Fractionating variables.

constraints. This same constraint would have two valid **RLT0** bound factor complements as multipliers: $(1-x_3)$ and $(1-x_5)$. Multiplication by these factors yields the two **RLT0** constraints

$$x_1 + x_2 + x_3 + x_4 \leq 1,$$

$$\text{and} \qquad x_1 + x_2 + x_4 + x_5 \leq 1,$$

both of which are dominated by their **RLT1** counterpart. Because of this strict dominance, and the observed ability of **RLT1** to recover the convex hull of integer feasible solutions for all test problems in this class, it follows that the weaker **RLT0** constraints produced in this case will be insufficiently tight to recover the convex hull, possibly leaving variables at fractional values that violate the tighter **RLT1** constraints.

For each of the test problems used in the spectrum of densities shown in Table 7, those possess-

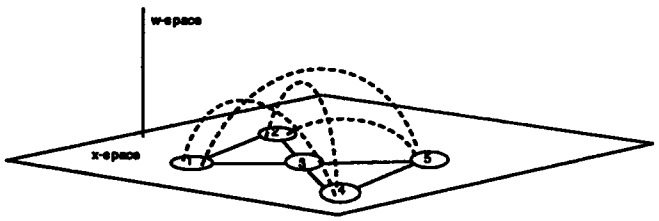

FIGURE 17. W-space adjacency for RLT.

ing an LP–IP gap were re-examined for evidence of the above behavior. The results are displayed in Table 9. Note the correspondence between the cancellation of w-variables and the performance of **RLT1** and **RLT0** in Table 8. This effect also identifies the conditions under which **RLT1** will recover the convex hull of binary solutions and **RLT0** will not. Precisely, when no w-variables survive a level-one application in some constraints of either **RLT1** or **RLT0**, and the density of the constraint coefficient matrix resides in the midranges, the reformulation generated by **RLT1** tends to produce a lower bound that strictly dominates that of **RLT0**.

5.0.6. Graph theoretical insights. There is an elegant graph theoretical relationship between adjacency and w-variable cancellation that helps to explain the strict dominance of **RLT1** over **RLT0** in such cases as noted above. Associate with each 0-1 variable x_j a node j, and an edge $[k, j]$ if node k and node j are adjacent in A, *i.e.*, x_k and x_j appear in some constraint together. Then, for any coefficient matrix A for the set packing problem, there exists an adjacency graph $G(N, E)$ with node set N, and edge set $E = \{[k, j] : k, j \in N_i, \bigcup_{i=1}^{m} N_i = N\}$ containing non-redundant adjacencies of A. This adjacency graph can then be represented in the space of original x-variables. For example, the adjacency existing within the two clique constraints $x_1 + x_2 + x_3 \leq 1$ and $x_3 + x_4 + x_5 \leq 1$ is represented in the figure of Example 8, Chapter 3, and is depicted in the x-space plane in Figure 17.

Now, notice that the only variables, w_{kj}, that survive adjacency cancellation resulting from a complete level-one application of RLT are those whose associated nodes k and j are not adjacent in G. Instead of letting the surviving w_{kj} represent a 'shrinking' of nodes k and j similar to the concept introduced by Johnson and Padberg (1982), we interpret w_{kj} as establishing an adjacency in the higher dimensional w-space between nodes k and j, effectively 'sewing' the nodes represented by x_k and x_j together. This effect is illustrated by the dashed lines in Figure 17, in

which redundant adjacencies present in the original coefficient matrix have been eliminated. This elimination is logically sound since only a single adjacency present in the constraint coefficient matrix is sufficient to cause cancellation of the associated w-variable, *i.e.*, if x_j and x_k appear together in a single constraint, then $w_{jk} = x_j x_k = 0$. Thus, the edges induced by the surviving level-one RLT w-variables correspond precisely to those edges not present in the original graph G. By denoting, as usual, $K[n]$ as the complete graph on n vertices, $E(K[n])$ as the edge set of $K[n]$, and $E(G)$ as the edge set of G, this leads to the following result.

Proposition 6. The RLT w-variables surviving any complete level-one application of either **RLT1** or **RLT0** to a set of constraints $Ax \leq e$, where e represents a vector of proper dimension whose components are all equal to one, are precisely those contained in the edge set $E(K[n]) - E(G)$, where $n = |\, N \,|$. ■

As a consequence of Proposition 6, an exact count of the number of unique w-variables that will appear in a complete level-one RLT reformulation follows directly from Proposition 6, and is given by the cardinality of the set $E(K[n]) - E(G)$.

Since the conditions under which adjacency cancellation occurs in the set packing constraints are rigidly enforced in set partitioning constraints $Ax = e$, the results apply to this additional class of problems. The following examples illustrate the application of Proposition 6. Note that, in an actual application to the problem class addressed herein, the adjacency information would be obtained by simply scanning the constraint coefficient data structure, and not by interposing such a graphical representation within the RLT process.

Example 1. The (w, x)-space representation on $K[5]$ for the set of constraints given by $x_1 + x_2 + x_3 \leq 1$ and $x_3 + x_4 + x_5 \leq 1$ is displayed in Figure 18. The higher dimensional adjacencies created by the w-variables surviving cancellation from a complete level-one application of RLT on these constraints, w_{14}, w_{15}, w_{24}, and w_{25} are shown in dashed lines. Additionally, the number of w-variables surviving adjacency cancellation is given by $|\, E(K[5]) - E(G) \,| = |10 - 6| = 4$. ■

Example 2. Consider the clique constraints given by $x_1 + x_2 + x_3 \leq 1$, $x_2 + x_4 + x_5 + x_6 \leq 1$, and $x_1 + x_6 + x_7 \leq 1$. The adjacency of the node set of these constraints is shown in Figure 19(a).

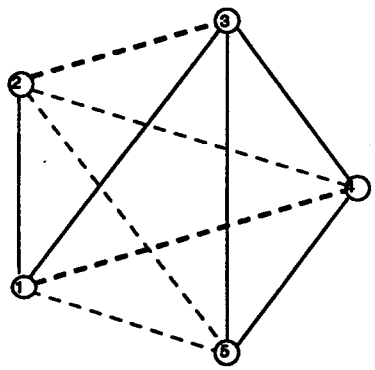

FIGURE 18. A (w,x)-space adjacency on K[5].

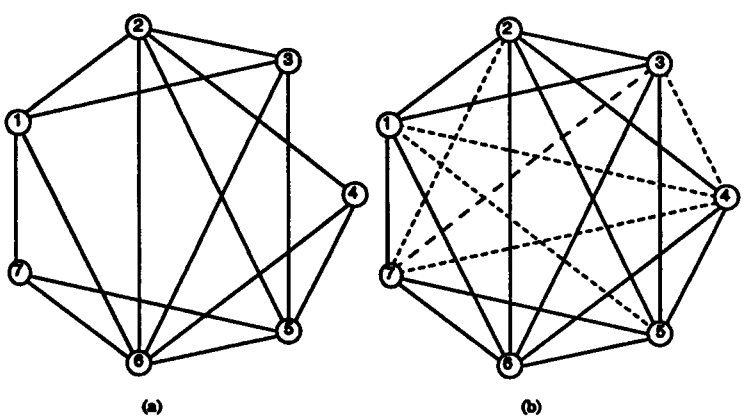

FIGURE 19. A (w,x)-space adjacency on K[7].

		MTZ		DLMTZ		RLT1MTZ		STN3ID		RLT13ID	
n	Z_{IP}	Z_{LP}	ITER	Z_{LP}	ITER	Z_{LP}	ITER	Z_{LP}	ITER	Z_{LP}	ITER
10	156	151.8	100	153.2	119	153.4	315	151.0	358	151.0	402
15	115	103.3	150	107.0	226	108.0	548	101.0	1336	101.0	884
20	125	116.8	184	117.1	356	118.2	595	116.0	3116	116.0	1394[a]
30	175	168.4	330	172.2	440	172.2	1913	168.0	10313[b]		[c]
40	204[d]	176.8	473	191.2	747	191.2	2468		[c]		[c]
50	160	153.1	451	155.1	1238	155.2	3809		[c]		[c]

TABLE 10. Lower bounds for ATSP formulations.

After applying a complete level-one application of RLT, the w-variables surviving adjacency cancellation are w_{14}, w_{15}, w_{27}, w_{34}, w_{37}, and w_{47}. In the (w,x)-space representation of Figure 19(b), these variables correspond precisely to the edge set $E(K[7]) - E(G)$. Augmenting the original adjacency edge set $E(G)$ with these higher dimensional edges yields the $K[7]$ representation shown in Figure 19(b). The number of w-variables surviving adjacency cancellation is given by $| E(K[7]) - E(G) | = |21 - 15| = 6$. ■

These graph theoretical results extend what was previously known for the class of constraints that contain generalized upper bounding (GUB), set packing, set partitioning, vertex packing, and maximal cardinality clique constraints. As stated in Example 3 of Chapter 3, the RLT level equal to the maximum number of variables that can simultaneously be 1 is known to be sufficient to recover the convex hull representation. Proposition 6 allows us to identify precisely from the original adjacency graph, $G(N,E)$, which of the w-variables will be present after a complete level-one application. Moreover, they enable the identification of RLT products that are unnecessary, and, most importantly, they make it possible to identify *a priori* whether or not **RLT1** will strictly dominate **RLT0** in a complete level-one application, since, as borne out by the computational experiments conducted, **RLT1** tends to strictly dominate **RLT0** whenever no w-variables survive and the density of the coefficient constraint matrix is in the mid-range of the spectrum.

5.0.7. Results for ATSP. Table 10 displays the number of cities defining each test problem (n), the corresponding integer optimal solution (Z_{IP}), the value of the linear programming re-

laxation (Z_{LP}), and the number of simplex iterations required to obtain the optimal solution for various sized ATSP problem instances using each of the five formulations presented in Chapter 4. The **RLT1** formulation of MTZ required a greater number of simplex iterations to achieve optimality in all cases. However, this **RLT1** formulation consistently performed at least as well as the formulation of Desrochers and Laporte (1991) (DL), and strictly dominated all other formulations of ATSP tested. This result directly supports the theoretical dominance of **RLT1** over the formulation proposed by Desrochers and Laporte (1991) as stated in Proposition 3. The TSP formulation with Miller, Tucker, Zemlin (MTZ) subtour elimination constraints provided better lower bounds than either of the two 3-index ATSP formulations, but consistently worse than the other MTZ formulations, confirming information reported in Padberg and Sung (1992), and Langevin, Soumis, and Desrosiers (1990).

Both of the 3-index formulations performed poorly on this set of test problems. The new **RLT1** constraint derived from the quadratic assignment formulation does not appear to enhance constraint tightening, since the lower bound found by RLT13ID simply matched that of the standard 3-index ATSP formulation. Additionally, difficulties were encountered while implementing these formulations, and several points are significant with reference to the entries of Table 10:

(a) The size of both 3-index ATSP data structures grew rapidly as the value of n increased. In this particular instance, 34 Mb of random access memory was required to accommodate the formulation.

(b) Although a lower bound was found for this larger instance, an extreme amount of 'tailing' occurred as the solution approached its reported optimum. Upward of 5000 simplex iterations were performed by the OSL solver to advance the LP relaxation solution from 171.4 to its reported value.

(c) The size of these problem instances exceeded memory limitations during the model generation phase, preventing analysis. The smallest formulation required 45 Mb of random access memory. The excess size of these formulations also precluded their use in the context of the three problems taken from TSPLIB. In comparison, an $n = 100$ problem instance was generated using the Desrochers and Laporte (1991) formulation, and it required only 9 Mb.

(d) The OSL branch-and-bound algorithm stopped after reaching the preset limit of 100,000 iterations. The algorithm evaluated a total of 6285 nodes without confirming an optimal solution. This test problem, henceforth known as `pjd40.atsp`, is being submitted to the TSPLIB for further analysis and testing.

It is not surprising that both of the 3-index TSP formulations rapidly exceeded the memory limitations imposed by the OSL solver. Both are variations of the time-dependent TSP, which has not been as extensively studied as the TSP. Moreover, the few computational studies available suggest that it is a more difficult problem to solve and exact approaches have been reported only for special cases. These approaches have involved instances with only up to a few tens of vertices (see Picard and Queyranne (1978), and Simchi-Levi and Berman (1991)). Only recently has a reasonable exact solution approach been proposed by Vander Wiel and Sahinidis (1994) based on an application of Benders' decomposition. However, the set of test problems used to demonstrate the effectiveness of this approach had $n \leq 15$ for all instances.

The three problems of Table 11 taken from the TSPLIB are known to be particularly difficult instances that challenge algorithms to solve them to optimality. Thus, they appeared to provide a good context within which to assess the bounds obtainable via the three MTZ formulations. By far, the weakest bounds were obtained from the usual MTZ formulation, again confirming its previously reported inability to close an existing LP–IP gap. The lower bound obtained by the **RLT1** formulation strictly dominates that of the other formulations for these TSPLIB instances, although it consistently used a greater number of simplex iterations to do so. A greater degree of efficiency could be realized by using a Lagrangian dual approach for solving the resulting **RLT1** reformulations, as suggested by Sherali and Adams (1994), thus pursuing an optimal solution over a smaller number of constraints than that present in the primal reformulation.

Additionally, we attempted to directly solve the IP version of TSPLIB problems `br17` and `p43` using these three formulations. Interestingly, the **RLT1** formulation found the reported optimal value of 39.0 for problem `br17` in 28 branch-and-bound nodes, whereas both MTZ and DLMTZ halted after exceeding the 100,000 iteration limit, reporting Z_{IP} values of 52.0 and 41.0, respectively. In the process of obtaining these weaker bounds, the MTZ formulation and the DLMTZ formulation required the OSL branch-and-bound algorithm to evaluate 2214 and 2522

TSPLIB	Z_{IP}	MTZ		DLMTZ		RLT1MTZ	
		Z_{LP}	ITER	Z_{LP}	ITER	Z_{LP}	ITER
br17	39	2.25	134	22.0	185	27.7	674
p43	2810	74.8	464	108.0	395	432.3	2538
ry48p	14422	12430.5	733	13577.8	720	13602.5	4249

TABLE 11. Results using TSPLIB test problems.

nodes, respectively. All three formulations were unable to locate an optimal integer solution prior to reaching iteration limits for problem p43, confirming the difficulty of this TSPLIB problem.

Although the three MTZ formulations are primarily intended for asymmetric TSP's, Lovasz and Schrijver (1991) reported the performance of DLMTZ on symmetric problems as well, hence, we also used report the performance of the **RLT1** formulation for symmetric versions of the problems shown in Table 10. The results are displayed in Table 12. Again, although **RLT1** used a greater number of simplex operations to reach optimality for each test problem instance, the lower bounds were consistently as tight, or tighter than that of the DLMTZ formulation. This again demonstrates that the dominance predicted by Proposition 3 can be strict.

The three formulations were also tested as MIPs using problem br17 and a symmetric TSP with $n = 50$. Table 13 displays the results of this experiment. In contrast to requiring a greater number of simplex iterations, the **RLT1** formulation requires a far smaller number of nodes to be examined to reach optimality within a branch-and-bound environment. This performance was also observed when solving a second symmetric TSP with $n = 50$ on a computer with significantly more memory. This second problem was solved as an MIP on a larger RS/6000 590 at the GAMS Development Corporation, using the OSL MIP solver within the AIX environment. For this problem, the OSL MIP solver reached optimality after evaluating only 21 branch-and-bound nodes, and neither the OSL solver, nor an additional CPLEX solver were able to obtain any further tightening than that afforded by the **RLT1** formulation. This performance is further evidence of the quality of the RLT1MTZ formulation not only as a relaxation, but also as an 0-1 MIP.

		MTZ		DLMTZ		RLT1MTZ	
n	Z_{IP}	Z_{LP}	ITER	Z_{LP}	ITER	Z_{LP}	ITER
10	161	146.2	51	161.0	117	161.0	278
15	131	93.8	107	125.0	194	127.4	414
20	184	150.1	218	179.0	322	179.0	767
30	215	153.1	374	200.0	622	200.1	2102
40	200	154.4	548	199.0	753	199.2	2742
50	212	179.0	559	211.5	797	211.5	3586

TABLE 12. Lower bounds for symmetric TSP formulations.

		MTZ		DLMTZ		RLT1MTZ	
PROBLEM	Z_{IP}	BEST	NODES	BEST	NODES	BEST	NODES
br17	39.0	52.0	2214	41.0	2522	39.0	28
n50	212	234.0	2863*	212.0	48	212.0	7

TABLE 13. Integer solutions for TSP formulations.

CHAPTER 6

Finale

The most promising of current techniques for solving 0-1 mixed-integer programs focus on algebraically manipulating the constraints of the model so that the geometric region defined by the constraints of the linear programming representation is as constricted, or as *tight*, as possible with respect to the smallest polyhedral geometric region that encapsulates all integer feasible solutions. If one were able to tighten the constraints to such a degree that the relaxed feasible region coincided with the integer feasible region, known as the convex hull of integer feasible solutions, the ideal would be achieved: solving the linear programming representation would solve the mixed-integer problem as well. Practicality and experience has dictated that a tightening process of this nature should be applied in degrees, depending upon the specifics of the problem being addressed.

6.1. Conclusions

The new hierarchy of relaxations developed in this research extends the Reformulation–Linearization Technique (RLT) of Sherali and Adams (1989, 1990), currently state-of-the-art in this field, and is particularly designed to exploit explicit or implicit special structures defined by the constraints of a problem. Moreover, in addition to recognizing existing special structures for developing specific factors in the RLT process, this hierarchy also subsumes the Sherali–Adams (1989, 1990) hierarchy as a special case. Inherent special structures are exploited by identifying specific classes of multiplicative factors that can be applied to the original mathematical formulation of the problem to initially reformulate it as an equivalent polynomial programming problem. Subsequently, this resulting problem is linearized to produce a tighter relaxation in a higher dimensional space cre-

ated by a substitution of variables. The hierarchy is defined by selecting *a priori* the number and type of multiplicative factors to be applied to the constraint set. The level of relaxation, equal to the highest degree of the polynomial terms used in the multiplicative factors is thus derived. This general framework permits us to generate an explicit hierarchical sequence of tighter relaxations leading up to the convex hull representation in which the final level equals the number of 0-1 integer variables involved in the problem. Although several other researchers have subsequently proposed reformulation-linearization techniques that are variations of the original Sherali-Adams (1989, 1990) technique, they remain, for the most part, existential in nature because of the difficulty in providing explicit algebraic representations that can be subsequently used in practice to derive other valid constraints that further assist in tightening the problem formulation. **RLT1** not only provides an explicit algebraic representation of a specific relaxation level, but additionally enables the relaxation to be solved in the higher dimensional space created, *without any* projection operation being performed. The equivalence of **RLT1** with the well-known sequential lifting technique introduced by Balas and Zemel (1978) was also demonstrated.

In addition to presenting the relevant theory, a complete level-one relaxation generated by this technique was applied in detail as an automatic reformulation scheme to various set packing problems of different sizes. Computational results using these reformulations validate the theoretical dominance of **RLT1** over **RLT0** and the ordinary LP relaxations associated with these instances. Morever, a graph theoretical interpretation of a complete level-one application of both **RLT1** and **RLT0** has identified the conditions under which the bound achieved by **RLT1** might strictly dominate that attained by **RLT0** for the same instance. In all the test cases of the set packing problem, however, it was found that *a complete level-one application of **RLT1** was sufficient to recover the integer optimal solution.*

RLT1 was also applied to several formulations of the asymmetric traveling salesman problem (ATSP). The motivation for doing so is that the ATSP is an important and classical combinatorial optimization problem that happens to possess rich exploitable special structures, and arises in diverse applications related to production planning, vehicle routing and scheduling. Moreover, many exact or heuristic techniques that have been developed for the solution of hard combinatorial optimization problems were originally conceived and tested on the TSP, which, as a result,

has become a prototype problem in this sense. By this application of the new hierarchy, new formulations have been developed and tested for the 3-index TSP, and the standard TSP formulation with Miller, Tucker, and Zemlin subtour elimination constraints. Although computational experimentation with the **RLT1** formulation of the 3-index TSP was disappointing, succeeding in only matching the bounds attainable using the standard 3-index TSP, the **RLT1** formulation that exploited the structure afforded by MTZ subtour elimination constraints confirmed the theoretical dominance demonstrated in Proposition 3. The bounds attained by this **RLT1** formulation were consistently as good or better than that available via the other formulations examined. This new formulation holds promise for successfully defeating larger problem instances within this class of hard combinatorial problems. In fact, for one such standard problem from the literature, the reformulated problem was able to detect an optimum within 28 branch-and-bound nodes, while the previous best MTZ tightened formulation was unable to do so after enumerating 2214 branch-and-bound nodes. The equivalence of **RLT1** and sequential lifting was further demonstrated in the context of lifting cycle inequalities resulting from subtours formed from solving the corresponding assignment problem of a particular TSP.

Lastly, additional ideas for further strengthening RLT–based constraints by using conditional logical implications, as well as relationships with the widely applied "sequential lifting process," were also presented. As is discussed in the sequel, these conditional logic strenthening ideas are envisioned to provide the cornerstone for implementing **RLT1** within a parallel processing environment. Several examples of classic special structures that arise in a host of combinatorial optimization problems were presented to demonstrate how underlying special structures, including generalized and variable upper bounding, covering, partitioning and packing constraints, as well as sparsity, can be exploited within this framework. These structures arise in various models of economic planning, production, personnel scheduling, distribution, and engineering design. For some types of structures, low level relaxations were exhibited to recover the convex hull of integer feasible solutions. We also introduced an alternative partial application strategy for this new hierarchy, designed to reduce the growth in number of constraints necessary to be added to the reformulation of a problem, that provides a continuum of strategies that can be applied between **RLT0** and **RLT1**.

From cited papers that have reported additional encouraging computational results for both linear and multilinear 0-1 mixed-integer problems using only partial ideas at level one of the proposed schemes, it appears that further enhancements using the suggested concepts presented herein might hold promise. While we have focused on the linear problem in this dissertation, the results directly extend to mixed 0–1 polynomial problems, for which a similar hierarchy leading to the convex hull representation can be generated. Furthermore, in the light of Sherali and Tuncbilek (1992, 1994), relaxations based on S-factors can also be evidently extended to solve nonlinear, nonconvex, polynomial programming problems.

As mentioned earlier, the importance of this research becomes apparent when one realizes that *all* commercially available software packages currently utilize the linear programming relaxation of a 0-1 mixed-integer problem to guide the search of branch-and-bound algorithms through the binary branching tree. As demonstrated in this dissertation, for certain applications of this new hierarchy, not only are several levels of this branching process resolved at the very first node, but an implicit pruning of the branching tree also occurs. Thus, for these applications, not only is the initial linear programming representation tightened, yielding sharper bounds on the objective function value, but the number of branches of the binary tree requiring to be searched is reduced. This fact was borne out in our computational tests as mentioned above on one difficult problem instance from the TSPLIB

Future research in this area will attempt to extend the tenets of the new hierarchy within a decomposition scheme so as to take advantage of parallel computing architecture. To do so requires that portions of the **RLT1** process be consolidated, and others distributed to various processors. A suggested general technique for accommodating this distribution is suggested in the sequel.

6.2. Extensions

6.2.1. Parallel implementation issues. In light of the fact that researchers are turning to parallel architectures in the hope of conquering larger and larger classes of mixed-integer programs and pure integer programs, some exposition is in order on how this new hierarchy might be implemented within a parallel computing environment. The choice of an implementation strategy

for applying **RLT1** is greatly influenced by the design of the parallel machine to be used, as well on certain inherent characteristics of **RLT1**. In consideration of the former, parallel machines differ in the manner in which memory is available to the individual processors. There are fundamentally two configurations currently in use: tightly coupled and loosely coupled parallel architectures. Tightly coupled, global memory multiprocessors that offer all processors equal access to shared common memory have a certain attractiveness based on the ease in which serial algorithms can be converted into parallel versions on these systems. However, the tight coupling along the critical path between individual processors and memory has its downside: *contention*. This contention relates to processor to memory interconnections, paths through the interconnections, and access to specific memory modules, and has an impact upon performance. As the number of processors accessing this global memory scales up, the performance of the system drops off from a linear response.

Despite these detractors, the appeal of shared memory systems is apparent because of its ability to accommodate different programming styles and even a mix of styles, so that existing single processor algorithms can readily be ported to parallel platforms with a minimum of difficulty. Programs written to specifically exploit the characteristics of a particular architecture, which is the case with loosely coupled systems, lose this appeal and currently require programmers to adopt new paradigms to take advantage of parallelism.

In loosely coupled architectures, private memory prevents processors from being able to read on another's memory, forcing the individual processors to pass data along some communications link to another processor. Communications links of this nature exist either as hard connections within a single machine, or via some local area switching network that links multiple machines together for a specific task. Since message passing is generally implemented at a subroutine level and requires cooperation at the receiving end, this is a source of efficiency loss for loosely coupled systems.

For mathematical programming algorithms that naturally lend themselves to partitioning of data and task subdivision, such as branch-and-bound procedures, an efficient approach to implementing **RLT1** would be to employ a loosely coupled system, minimizing the amount of information that must be passed between processors. Additionally, distributing the reformulation to

different memory banks reduces the degree of program and data locality and enhances memory reference patterns that are more uniformly distributed. This reduces the contention for memory modules and *tree saturation* caused by processors trying to access 'hot' memories, and blocking switches in the interconnections that are holding traffic in a queue. Uniformly distributing data balances the system load over the available modules.

The **RLT1** hierarchical approach to tightening linear programming relaxations possesses several aspects of parallelism that could be exploited within a branch-and-bound parallel processessing environment. In a complete level-one application for instance, after identifying a suitable special structure to compose the set S, a pairwise multiplication of each constraint and each factor must be executed, a task amenable to distributed processing much like that used by Boehning, Butler and Gillett (1988) for analyzing new nodes in the branching tree. Additionally, the task of performing repeated conditional logic tightening of **RLT1** constraints lends itself well to parallel implementation, since the variable fixing that motivates such tightening need not occur in a centralized location as long as the tightening information is subsequently distributed. This would insure that the cascading effect of one processor's variable fixing efforts onto another is not lost due to isolation.

It is not envisioned that **RLT1** will be implemented on a global shared memory system. This option takes least advantage of the potential power of parallel computations. Providing a single reformulation of the original problem, storing it in global memory and then decomposing the problem to distribute various partitions to different processors in a manner already accomplished with branch-and-bound, does little beyond simply transferring a sequential process to a multiprocessor environment. Each of the processors would simultaneously apply standard cutting planes, variable fixing, and preprocessing as is current used.

Unquestionably, if a loosely coupled distributed system is chosen, the amount of subproblem comparison and exchange of information that occurs between processors should be kept to a minimum to avoid contention. In the desciption that follows, we assume some variation of depth-first search is being utilized by the branch-and-bound algorithm, since it requires less storage overall. A purely best-first search performed concurrently on different processors implies a localization of search space information, preventing the best bound obtained by one processor from helping to

prune the search space of another. **RLT1** naturally supports variations of a depth-first search strategy because of the constraint tightening available via exploiting special structures. If the appropriate level of relaxation is chosen, the bound obtained from the initial **RLT1** reformulation could be close to the integer optimum. By using a depth-first search, a large portion of the search space can easily be pruned as a result. This type of effect was seen when solving the TSPLIB test problem `br17` to optimality.

RLT1 could operate within several parallel implementations that differ in the manner in which problem distribution occurs. One implementation divides the search space into several disjoint parts, each of which is searched concurrently in a depth-first manner by a different branch-and-bound process. A second implementation has multiple processors performing depth-first search concurrently. With this second distribution scheme, at any time, at least one process has the property that if it terminates, it returns an optimal solution; the other processes conduct the standard look-ahead search. Both of these implementations minimize the amount of communication occurring between processors, and hence, are ideally suited for a loosely coupled system. The first implementation might be less attractive for **RLT1** since it naturally inhibits conditional logic tightening from occurring beyond the initial reformulation. Therefore, we describe a technique for adapting **RLT1** within the second implementation.

At the onset, an initial reformulation of the problem at-hand is constructed in an automatic manner. The pairwise multiplication of constraints is a type of fine grained parallelism that can easily be divided and distributed to the various processors in a master–slave mode. After this step is complete, standard preprocessing would be performed on the complete reformulation. When no further progress in this vein is made, the resulting reformulation is solved as a linear programming relaxation. Suppose that a number of binary variables fractionate in this initial solution. With multiple processors, several branch-and-bound processes can subsequently be started, one on each processor, and each with a different starting bound. To speed up the search, the bounds on all but one processor are started optimistically closer to the expected optimal value in the following manner. Each processor fixes the value of a different fractional variable at one of its bounds. Each processor initialized in this manner can begin independent conditional logic tightening of **RLT1** constraints based on the fixed variable, which, in turn, can accommodate a selective reformulation

to occur at each processor. The one remaining processor is left with the most 'pessimistic' bound, that obtained by the initial relaxation, in order to guarantee possession of an optimal solution.

Optionally, using a master program to control the problem distribution to the various processors, it is also possible to simply use the multiple processors to simultaneously solve several LP relaxations, and return optimality information to the main processor, which subsequently makes branching and fathoming decisions based on this information. This strategy could be implemented on a Local Area Network (LAN) composed of separate workstations by using the program Parallel Virtual Machine (PVM) available in NETLIB, combined with the object environment of FORTRAN 90. Defining specific aspects of the **RLT1** process as objects, and passing these objects to the various processors enables a programmer to define the objects containing these **RLT1** processes only once, and allows the master program to control any distribution of these processes during execution.

This implementation mimimizes contention by requiring very little communication between individual processors. Message passing occurs only when a better solution is found that improves upon the one previously identified. The criteria used for allowing subsequent fixed variable information to be passed is if the binary value of a particular variable is determined to persist in any optimal solution. The conditions under which this would occur in the context of RLT relaxations were identified recently by Adams, Lassiter, and Sherali (1993), and are dependent upon the corresponding optimal dual multipliers. Moreover, the independent nature of the distribution precludes the processors from having to wait for information, thus allowing them to continue or terminate according to whatever information is currently available.

One drawback to implementing **RLT1** within this distribution scheme is the potentially large amount of information that is initially passed to each processor. A major difficulty arising due to the size of the **RLT1** reformulation occurs because of requiring the problem to be represented in its primal form. The reformulation could be passed to each processor in some implicit aggregated form amenable to Lagrangian dual optimization, which would help alleviate the problem. Additionally, as suggested by Pekny and Miller (1992), solved problems could be placed in a queue that can be cached to secondary storage using a replacement strategy based on bound information. In this sense, only a very few of these solved problems written to secondary storage are retrieved. Hence,

only a small fraction of the aggregate search tree need reside in local memory at any one time.

Conditional logic tightening past the intial preprocessing might be accommodated by setting pointers to specific **RLT1** multiplier data, and updating would be performed as required. The tightened constraint information could then be either passed to the remainder of processors, or stored in cached memory for access. Care must be taken to avoid creating a 'hot' memory bank that all processors would be attempting to access often, thus exacerbating the problem of contention experienced in loosely connected system. Care must also be taken to insure that the local information exchanged between processors does not cause discontinuous jumps from one area of a search domain to another that might inadvertently miss better solutions.

6.2.2. Future research directions. As was stated in Chapter 5, although the presentation of **RLT1** is complete within this dissertation, our intention is that it also serves as a starting point for future research efforts. With this in mind, there are several directions that future research could take to extend the tenets of this new hierarchy, and resolve implications of the results of this research.

The results obtained by applying **RLT1** to the TSPLIB problem `br17` motivate further attempts to tighten the formulation of p43 using automatic **RLT1** constraints other than "cutting planes." In particular, we intend to investigate an automatic reformulation using **RLT1** products derived from subtours formed by solving the LP relaxation of p43 over the assignment constraints. Karp (1979) presented an algorithm for the approximate solution of the ATSP that first solves the assignment problem over the cost matrix, and then patches the subtour cycles together of the optimum assignment together to form a tour. Using an appropriate level-d **RLT1** application to these subtours, we intend to determine if it is possible to develop a means of artificially connecting subtours resulting from the assignment constraints that differs from the technique of Karp.

We intend to apply the technique of automatic reformulation to larger classes of problems, in order to determine if the pattern of connectivity-dependent performance identified in Chapter 5 continues to hold for general coeffient matrices for constraints of the form $\sum_{j=1}^{n} a_j x_j \leq a_0$ with either $a_j \equiv 1$ forall $j = 1, \ldots n$, and $a_0 \neq 1$, or $a_j \in Z^+$ forall $j = 1, \ldots, n$ and $a_0 \neq 1$. We would like to answer the question of whether the difference in performance between **RLT1** and **RLT0** remains dependent upon the appearance of w-variables, and fractionating x-variables for

these classes of constraints. Additionally, the existing coefficient matrices for the set packing test problems will be reexamined using an objective measure of connectivity. A scatter-plot of the effectiveness of **RLT1** and **RLT0** versus this connectivity measure should display the conceptualization pictured in Figure 16.

To take advantage of many advanced preprocessing procedures, we intend to develop a link between the automatic reformulation afforded by GAMS to the software program MINTO developed at Georgia Tech in order to test similar data structures as those used in this study with techniques not available within the OSL solver. In particular, we would like to have the ability to switch 'on' and 'off' cutting plane generation, logical preprocessing, and other capabilities of MINTO to examine their effect on the reformulations presented herein.

Lastly, **RLT1** produces algebraic representations of a particular problem in higher dimensions that could permit the derivation of valid inequalities and classes of facets for specific types of problems when projected onto the original variable space. This approach can be studied for different types of problems, as was accomplished by Sherali, Lee, and Adams (1992) for the Boolean Quadric Polytope.

CHAPTER 7

References

ADAMS, W.P., J.B. LASSITER, AND H.D. SHERALI. Manuscript 1993. Persistency in 0-1 optimization. Research Report, Department of Mathematical Sciences, Clemson University, Clemson, South Carolina.

ADAMS, W.P., AND H.D. SHERALI. 1986. A tight linearization and an algorithm for zero–one quadratic programming problems. *Mgmt. Sci.* **32**, 1274–1290.

ADAMS, W.P., AND H.D. SHERALI. 1990. Linearization strategies for a class of zero–one mixed integer programming problems. *Oper. Res.* **38**, 217–226.

ADAMS, W.P., AND H.D. SHERALI 1993. Mixed–integer bilinear programming problems. *Math. Prog.* **59**, 279–305.

BALAS, E. 1965. An additive algorithm for solving linear programs in 0-1 variables. *Oper. Res.* **13**, 519–549.

BALAS, E. 1975. Facets of the knapsack polytope. *Math. Prog.* **8**, 146–164.

BALAS, E. 1983. Disjunctive programming and a hierarchy of relaxations for discrete optimization problems. *SIAM J. Alg. Disc. Meth.* **6**, 466–486.

BALAS, E., S. CERIA, AND G. CORNUÉJOLS. 1993. A lift–and–project cutting plane algorithm for mixed 0–1 programs. *Math. Prog.* **58**, 295–324.

BALAS, E., S. CERIA, AND G. CORNUÉJOLS, G. PATAKI. 1994. Polyhedral methods for the maximum clique problem. Research Report, Graduate School of Industrial Administration, Carnegie Mellon University, Pittsburgh, Pennsylvania.

BALAS, E., AND E. ZEMEL. 1978. Facets of the knapsack polytope from minimal covers. *SIAM*

J. App. Math. **23**, 119–148.

BEALE, E.M.I. 1979. Branch and bound methods. *Ann. Disc. Math.* **5**, 201–219.

BENDERS, J.F. 1962. Partitioning procedures for solving mixed-variables programming problems. *Numer. Math.* **4**, 238–252.

BOEHNING, R.L., R.M. BUTLER, AND B.E. GILLETT. 1988. A parallel integer linear programming algorithm. *Eur. J. Oper. Res.* **34**, 393–398.

BOWMAN, V.J., AND G. NEMHAUSER. 1971. Deep cuts in integer programming. *Oper. Res.* **8**, 89–111.

BOYD, E.A. 1994. Fenchel cutting planes for integer programs. *Oper. Res.* **42**, 53–64.

BRADLEY, G.H., P.L. HAMMER, AND L. WOLSEY. 1974. Coefficient reduction for inequalities in 0-1 variables. *Math. Prog.* **7**, 263–282.

BROOKE, A., D. KENDRICK, AND A. MEERAUS. 1992. *The General Algebraic Modeling System (GAMS) User's Guide.* The Scientific Press, San Francisco, California.

CHVÁTAL. 1973. Edmonds polytopes and a hierarchy of combinatorial problems. *Disc. Math.* **4**, 305-337.

CROWDER, H., AND M. PADBERG. 1980. Solving large-scale traveling salesman problems to optimality. *Mgmt. Sci.* **26**, 495–509.

CROWDER, H., R.S. DEMBO, AND J.M. MULVEY. 1979. On reporting computational experiments with mathematical software. *ACM Trans. Math. Soft.* **5**, 193–203.

CROWDER, H., E.L. JOHNSON, AND M. PADBERG. 1983. Solving large-scale zero-one linear programming problems. *Oper. Res.* **5**, 803–834.

DAKIN, R.J. 1965. A tree-search algorithm for mixed integer programming problems. *The Computer Journal* **8**, 250–255.

DANTZIG, G., D. FULKERSON, AND S. JOHNSON. 1954. Solution of a large scale traveling salesman problem. *Oper. Res.* **2**, 393–410.

DESROCHERS, M., AND G. LAPORTE. 1991. Improvements and extensions to the Miller–Tucker–Zemlin subtour elimination constraints. *Oper. Res. Ltrs.* **10**, 27–36.

DRIEBEEK, N.J. 1966. An algorithm for the solution of mixed integer programming problems. *Mgmt. Sci.* **12**, 576–587.

DIETRICH, B.L., AND L.F. ESCUDERO. 1990. Coefficient reduction for knapsack-like constraints in 0-1 programs with variable upper bounds. *Oper. Res. Ltrs.* **9**, 9–14.

DIETRICH, B.L., AND L.F. ESCUDERO. 1992. On tightening cover induced inequalities. *Eur. J. Oper. Res.* **60**, 335–343.

DIETRICH, B.L., L.F. ESCUDERO, AND F. CHANCE. 1993. Efficient reformulation for 0-1 programs - methods and computational results. *Disc. Appl. Math.* **42**, 147–175.

EDMONDS, J. 1965. Maximum matching and a polyhedron with 0, 1 vertices. *J. Res.*, National Bureau of Standards Rpt 69B, 125.

EPPEN, G.D., AND K.R. MARTIN. 1987. Solving multi-item capacitated lot sizing problems using variable redefinition. *Oper. Res.* **35**, 832–848.

FOX, K. 1973. Production scheduling on parallel lines with dependencies. Ph.D. Thesis, Johns Hopkins University, Baltimore, Maryland.

FOX, K., B. GAVISH, AND S. GRAVES. 1980. An n-constraint formulation of the (time dependent) traveling salesman problem. *Oper. Res.* **28**, 1018–1021.

FREEMAN, R.J. 1965. Computational experience with the Balas integer programming algorithm. The Rand Corporation, P-3241.

FORTET, R. 1959. L'algèbre de Boole et ses applications en recherche opérationnelle. *Cahiers Centre Etudes Recherche Oper.* **1**, 5–36.

FORTET, R. 1960. Applications de l'algèbre de Boole en recherche opérationnelle. *Rev. Francaise Informat. Recherche Oper.* **4**, 17–26.

GENDRION, B., AND T.G. CRAINIC 1994. Parallel branch-and-bound algorithms: survey and synthesis. *Oper. Res.* **42**, 1042–1066.

GEOFFRION, A.M. 1969. An improved implicit enumeration approach for integer programming. *Oper. Res.* **17**, 437–454.

GLOVER, F. 1975. Improved linear integer programming formulations of nonlinear integer programs. *Mgmt. Sci.* **22**, 455–460.

GLOVER, F., AND E. WOOLSEY. 1974. Converting the 0–1 polynomial programming problem to a 0–1 linear program. *Oper. Res.* **22**, 180–182.

GLOVER, F., AND R.E. WOOLSEY. 1973. Further reduction of zero-one polynomial programm-

ming problems to zero-one linear programming problems. *Oper. Res.* **21**, 1, 156–161.

GOEMANS, M. 1989. Valid inequalities and separation for mixed 0-1 constraints with variable upper bounds. *Oper. Res. Ltrs.* **8**, 315–322.

GOMORY, R.E. 1960. An algorithm for the mixed integer problem. The Rand Corporation, RM-2597.

GOMORY, R.E. 1963. An algorithm for integer solutions to linear programs, in: Graves and Wolfe, eds., Recent Advances in Mathematical Programming, 269–302.

GREENBERG, H.J. 1990. Computational testing: why, how, and how much. *ORSA J. Comp.* **2**, 94–97.

GRÖTSCHEL, M. 1977. *Polyedrische Charakterisierungen Kombinatorischer Optimierungsprobleme.* Hain, Meisenheim am Glan.

GRÖTSCHEL, M., AND M. PADBERG. 1975. Partial linear characterizations of the asymmetric travelling salesman polytope. *Math. Prog.* **8**, 378–381.

GRÖTSCHEL, M., AND M. PADBERG. 1979. On the symmetric traveling salesman problem II: liftings and facets. *Math. Prog.* **16**, 281–302.

GONDRAN, M. 1979. Report on the session on cutting planes. *Ann. Disc. Math.* **5**, 193–194.

GUIGNARD, M., K. SPIELBERG, AND U. SUHL. 1978. Survey of enumeration methods for integer programming. In *Proceedings, SHARE 51 ACM*, Boston, 2161–2170.

HOFFMAN, K., AND M. PADBERG. 1985. LP–based combinatorial problem solving. *Ann. Oper. Res.* **4**, 145–194.

HOFFMAN, K.L., AND M. PADBERG. 1991. Improving LP-representations of zero-one linear programs for branch-and-cut. *Oper. Res.* **3**, 121–134.

HAMMER, P.L., E.L. JOHNSON, AND U.M. PELED. 1975. Facets of regular 0-1 polytopes. *Math. Prog.* **8**, 129–206.

HANSEN, P. 1979. Methods of nonlinear 0-1 programming. *Ann. Disc. Math.* **5**, 53–70.

JANAKIRAM, V.K., E.F. GEHRINGER, D.P. AGRAWAL AND R. MEHROTRA. 1988. A randomized parallel branch-and-bound algorithm. *Inter. J. Paral. Prog.* **17**, 277–301.

JOHNSON, E.L., M.M. KOSTREVA, AND U. SUHL. 1985. Solving 0-1 integer programming problems arising from large scale planning models. *Oper. Res.* **33**, 803–819.

JOHNSON, E.L., AND M. PADBERG. 1982. Degree–two inequalities, clique facets, and biperfect graphs. *Ann. Disc. Math.* **16**, 169–187.

KARP, R.M. 1979. A patching algorithm for the nonsymmetric traveling salesman problem. *SIAM J. Comput.* **8**, 561–573.

KETTANI, O., AND M. ORAL. 1990. Equivalent formulations of nonlinear integer problems for efficient optimization. *Mgmt. Sci.* **36**, 115–119.

KINDERVATER, G.A.P., AND H.W.J.M. TRIENEKENS. 1988. Experiments with parallel algorithms for combinatorial problems. *Eur. J. Oper. Res.* **33**, 65–81.

KUDVA, G.K., AND J.F. PEKNY. 1993. A distributed exact algorithm for the multiple resource constrained sequencing problem. *Ann. Oper. Res.* **42**, 25–54.

KUMAR, V., AND L.N. KANAL. 1984. Parallel branch-and-bound formulatins for the AND/OR tree search. *IEEE Trans. Pattern Anal. and Mach. Intel.* **PAMI-6**, 768–778.

LANGEVIN, A., F. SOUMIS, AND J. DESROSIERS. 1990. Classification of traveling salesman problem formulations. *Oper. Res. Ltrs.* **9**, 127–132.

LAWLER, E., J. LENSTRA, A. KAN, AND D. SHMOYS. 1992. *The Traveling Salesman Problem*, John Wiley and Sons, New York, New York.

LAWLER, E.L., AND D.E. WOOD. 1966. Branch-and-bound methods: a survey. *Oper. Res.* **14**, 699–719.

LIN, S., AND B.W. KERNIGHAN. 1973. An effective heuristic algorithm for the travelling salesman problem. *Oper. Res.* **21**, 498–516.

LITTLE, J.D.C., K.G. MURTY, D.W. SWEENEY, AND C. KAREL. 1963. An algorithm for the traveling salesman problem. *Oper. Res.* **11**, 972–989.

LOVÁSZ, L., AND A. SCHRIJVER. 1991. Cones of matrices and set functions, and 0–1 optimization. *SIAM J. Opt.* **1**, 166–190.

MARTIN, G.T. 1966. Solving the travelling salesman problem by integer linear programming. *CEIR*, New York.

MARTIN, K.R. 1987. Generating alternative mixed-integer linear programming models using variable redefinition. *Oper. Res.* **35**, 820–831.

MARTIN, K.R., AND L. SCHRAGE. 1985. Subset coefficient reduction cuts for 0-1 mixed-integer

programming. *Oper. Res.* **33**, 505–526.

MILLER, D.L., AND J.F. PEKNY. 1989. Results from a parallel branch and bound algorithm for the asymmetric traveling salesman problem. *Oper. Res. Ltrs.* **8**, 129–135.

MILLER, D.L., AND J.F. PEKNY. 1993. The role of performance metrics for parallel mathematical programming algorithms. *ORSA J. Comp.* **5**, 26–28.

MILLER, C., A. TUCKER, AND R. ZEMLIN. 1960. Integer programming formulation of traveling salesman problems. *J. ACM* **7**, 326–329.

MITRA, G. 1973. Investigation of some branch and bound strategies for the solution of mixed integer linear programs. *Math. Prog.* **4**, 155–170.

NEMHAUSER, G.L., AND L.E. TROTTER. 1974. Properties of vertex packing and independence systems polyhedra. *Math. Prog.* **6**, 48–61.

NEMHAUSER, G., AND L. WOLSEY. *Integer and Combinatorial Optimization*, John Wiley and Sons, New York, New York, 1988.

ORAL, M., AND O. KETTANI. 1992. Reformulating nonlinear combinatorial optimization problems for higher computational efficiency. *Eur. J. Oper. Res.* **58**, 236–249.

PADBERG, M. 1973. On the facial structure of set packing polyhedra. *Math. Prog.* **5**, 199–215.

PADBERG, M. 1979. Covering, packing and knapsack problems. *Ann. Disc. Math.* **4**, 265–287.

PADBERG, M., AND S. HONG. 1977. On the travelling salesman problem: a computational study. Thomas J. Watson Research Center Report, IBM Research.

PADBERG, M., AND M.R. RAO. 1974. The travelling salesman problem and a class of polyhedra of diameter two. *Math. Prog.* **7**, 32–45.

PADBERG, M., AND G. RINALDI. 1987. Optimization of a 532-city symmetric traveling salesman problem by branch and cut. *Oper. Res. Ltrs.* **6**, 1–7.

PADBERG, M., AND G. RINALDI. 1991. A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM Rev.* **33**, 1, 60–100.

PADBERG, M., AND T. SUNG. 1992. An analytical comparison of different formulations of the traveling salesman problem. *Math. Prog.* **52**, 315–357.

PEKNY, J.F., AND D.L. MILLER. 1992. A parallel branch-and-bound algorithm for solving large asymmetric traveling salesman problems. *Math. Prog* **55**, 17–33.

PETERSEN, C. 1971. A note on transforming the product of variables to linear form in linear programs. Working Paper, Purdue University, Hammond, Indiana.

PICARD, J.C., AND M. QUEYRANNE. 1978. The time-dependent traveling salesman problem and its application to the tardiness problem in one machine scheduling. *Oper. Res.* **26**, 86–110.

POLLATSCHEK, M.A. 1970. Algorithms on finite weighted graphs. Ph.D. Thesis, Technion-Israel Institute of Technology, Faculty of Industrial and Management Engineering.

PRUUL, E. 1975. Parallel processing and a branch-and-bound algorithm. M.S. Thesis, School of Operations Research and Industrial Engineering, Cornell University.

SHERALI, H.D. AND W.A. ADAMS. 1986. A tight linearization and an algorithm for zero-one quadratic programming problems. *Mgmt. Sci.* **32**, 1274–1290.

SHERALI, H.D., AND W.P. ADAMS. Manuscript 1989. A hierarchy of relaxations and convex hull characterizations for mixed-integer zero-one programming problems. Research Report, Virginia Polytechnic Institute and State University, Blacksburg, Virginia.

SHERALI, H.D., AND W.P. ADAMS. 1990. A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems. *SIAM J. Disc. Math.* **3**, 411–430.

SHERALI, H.D., AND W.P. ADAMS. 1994. A hierarchy of relaxations and convex hull characterizations for mixed–integer zero–one programming problems. *Disc. Appl. Math.* **52**, 83–106. (Manuscript 1989)

H. SHERALI, AND W. ADAMS. 1993. Mixed–integer bilinear programming problems. *Math. Prog.* **59**, 279–305.

SHERALI, H.D., AND W.P. ADAMS. 1994. A reformulation–linearization technique (RLT) for solving discrete and continuous nonconvex programming problems. *Mathematics Today*, Vol XII-A, special issue on *Recent Advances in Mathematical Programming*, ed. O. Gupta, 61–78

SHERALI, H.D., W.P. ADAMS, AND P.J. DRISCOLL. 1994. Exploiting special structures in constructing a hierarchy of relaxations for 0-1 mixed integer problems. Research Report, Department of Industrial and Systems Engineering, Virginia Tech, Blacksburg, Virginia.

SHERALI, H.D., AND A. ALAMEDDINE. 1992. A new reformulation–linearization technique (RLT) for bilinear programming problems. *J. Global Opt.* **2**, 379–410.

SHERALI, H.D., AND Y. LEE. Manuscript 1992. Tighter representations for set partitioning problems via a reformulation–linearization technique. To appear in *Disc. App. Math.*

SHERALI, H.D., AND Y. LEE. Manuscript 1992. Sequential and simultaneous liftings of minimal cover inequalities for GUB constrained knapsack problems. To appear in *SIAM J. Disc. Math.*

SHERALI, H.D., Y. LEE, AND W.P. ADAMS. Manuscript 1992. A simultaneous lifting strategy for identifying new classes of facets for the boolean quadric polytope. To appear in *Oper. Res. Ltrs.*

SHERALI, H.D., AND C. TUNCBILEK. 1992. A global optimization algorithm for polynomial programming problems using a reformulation–linearization technique. *J. Global Opt.* **2**, 101–112.

SIMCHI-LEVI, D., AND O. BERMAN. 1991. Minimizing the total flow time of n jobs on a network. *IIE Trans.* **23**, 236–244.

TAHA, H.A. 1970. A balasian-based algorithm for 0-1 polynomial programming. Research Report No. 70-2, University of Arkansas, Fayetteville, Arkansas.

TOMLIN, J.A. 1971. An improved branch and bound method for integer programming. *Oper. Res.* **19**, 1070–1075.

TORRES, F.E. 1991. Linearization of mixed-integer products. *Math. Prog.* **49**, 427–428.

TRELEAVEN, P.C., D.R. BROWNBRIDGE, AND R.P. HOPKINS. 1982. Data-driven and demand-driven computer architecture. *Comput. Surveys* **14**, 93–143.

TROTTER, L.E. JR. 1975. A class of facet producing graphs for vertex packing polyhedra. *Disc. Math.* **12**, 373–391.

VAN ROY, T.J., AND L.A. WOLSEY. 1987. Solving mixed integer programming problems using automatic reformulation. *Oper. Res.* **1**, 45–57.

VANDER WIEL, R.J., AND N. SAHINIDIS. Manuscript 1994. An exact solution approach for the time-dependent traveling salesman problem. Research Report, Department of Mechanical and Industrial Engineering, University of Illinois, Urbana, Illinois.

WATSON, L.T., AND A.P. MORGAN. 1992. Polynomial programming using multi-homogeneous polynomial continuation. *J. Comp. Appl. Math.* **43**, 373–382.

WATTERS, L.G. 1967. Reduction of integer polynomial problems to zero-one linear programming problems. *Oper. Res.* **15**, 1171–1174.

WOLSEY, L.A. 1975. Faces of linear inequalities in 0-1 variables. *Math. Prog.* **8**, 165–178.

WOLSEY, L.A. 1976. Facets and strong valid inequalities for integer programs. *Oper. Res.* **24**, 367–372.

WOLSEY, L.A. 1990. Valid inequalities for 0–1 knapsacks and MIP with generalized upper bound constraints. *Disc. App. Math.* **29**, 251–261.

ZANGWILL, W.L. 1965. Media selection by decision programming. *J. Advertising Res.* **5**, 30–36.

ZEMEL, E. 1978. Lifting the facets of zero-one polytopes. *Math. Prog.* **15**, 268–277.

ZEMEL, E. 1989. Easily computable facets of the knapsack polytope. *Math. Oper. Res.* **14**, 760–764.

CHAPTER 8

Appendix

8.1. GAMS code for set packing problems.

```
$TITLE Vertex Packing Problem
$OFFUPPER
$OFFSYMXREF OFFSYMLIST OFFUELLIST OFFUELXREF
SETS
  I  clique constraint counter     /1 * 15 /
  J  variable index set            /1 * 25 /;
OPTION SEED = 5611;
OPTION RESLIM = 100000;
OPTION ITERLIM = 100000;
OPTION OPTCR = 0.0;
ALIAS(J1,J); ALIAS(J2,J); ALIAS(I1,I); ALIAS(I2,I);
SCALARS   rhs        /1/
          accept     /0.05/;
PARAMETER VLP  The value of the lp relaxation;
PARAMETER VIP  The value of the integer program;
PARAMETER RELGAP  The LP-IP percent gap;
PARAMETER ACTGAP  The actual LP-IP gap;
PARAMETER FRAC  The number of fractionating variables in solution;
PARAMETER BIN  The number of binary  x-variables in the solution;
```

```
PARAMETER   C(J)    cost vector of objective function ;
            C(J) = 1 + FLOOR( UNIFORM(0,99));
PARAMETER   A(I,J)  random constraint matrix;
            A(I,J) = UNIFORM(0,1);
DISPLAY "A AFTER ASSIGNING UNIFORM(0,1)", A;
   A(I,J) = 1 $(A(I,J) LE accept);
DISPLAY "A AFTER ACCEPTANCE FACTOR IS APPLIED", A;
VARIABLES
   W(J1,J)  Rlt variables for bound factors
   X(J)     vertex set
   Z       total cost scalar;
BINARY VARIABLE X;
POSITIVE VARIABLE W;
EQUATIONS
   COST             define objective function
   RLT(I,J1)        RLT nonnegative bound factor constraints
   RLTCOMP(I,J1)    RLT complementary bound factor constraints
   CLIQUES(I)       defining clique constraints
   CLIQUERLT(I,I2) RLT1 clique pairwise products;
   COST ..          Z =E= SUM ( J, C(J) * X(J) );
   CLIQUES(I) ..    SUM( J, A(I,J) * X(J))  =L= rhs;
   RLT(I,J1)$(A(I,J1) EQ 0) ..  SUM(J$(A(I,J) GT 0), ( A(I,J)*W(J1,J) )$
            (PROD(I1, A(I1,J1)*A(I1,J) EQ 0) AND ORD(J1) LT ORD(J) )  )  +
            SUM(J$(A(I,J) GT 0),( A(I,J)*W(J,J1))$
            (PROD(I1, A(I1,J1)*A(I1,J) EQ  0) AND  ORD(J1) GT ORD(J)) )
            =L=  (rhs*X(J1));
   RLTCOMP(I,J1)$(A(I,J1) EQ 0) .. ( SUM( J, A(I,J) * X(J) )) -
               (SUM(J$(A(I,J) GT 0), ( A(I,J)*W(J1,J))$
               (PROD(I1, A(I1,J1)*A(I1,J) EQ 0)  AND ORD(J1) LT ORD(J) ) )
```

```
                + SUM(J$(A(I,J) GT 0),(A(I,J)*W(J,J1))$
                (PROD(I1, A(I1,J1)*A(I1,J) EQ 0)  AND  ORD(J1) GT ORD(J)) )  )
                =L=  (rhs*(1-X(J1)));
$ONTEXT
This next set of constraints writes a clique RLT1 constraints in
the order that they are created:
$OFFTEXT
            CLIQUERLT(I,I2)$(ORD(I) LT CARD(I) AND ORD(I) LT ORD(I2)) ..
$ONTEXT
This sum simply lists the variables appearing in the clique being
multiplied.
$OFFTEXT
            SUM(J, A(I,J)*X(J)) +
$ONTEXT
This sum writes the variables from the multiplier clique only if
they do not appear in the clique being multiplied, regardless of
whether cancellation occurs for W variables.
$OFFTEXT
            SUM(J$(A(I2,J) GT 0), A(I2,J)*X(J)$(A(I,J)*A(I2,J) EQ 0) )  -
$ONTEXT
These two sums handle the generation of W variables based on
the conditions of RLT.
$OFFTEXT
            SUM( (J,J1)$(A(I,J) GT 0 AND A(I2,J1) GT 0),A(I,J)*W(J1,J)$
            (PROD(I1, A(I1,J)*A(I1,J1) EQ 0)  AND ORD(J1) LT ORD(J)) ) -
            SUM( (J,J1)$(A(I,J) GT 0 AND A(I2,J1) GT 0),
A(I,J)*W(J,J1)$(PROD(I1,A(I1,J)*A(I1,J1) EQ 0) AND ORD(J1) GT ORD(J))) =L= rhs;
$ONTEXT
The declaration of the following models allows them to be switched 'on' and
```

```
'off' as desired.
$OFFTEXT
MODEL   VERTLP      /COST, CLIQUES/
        VERTIP      /COST, CLIQUES/
        VERTRLT0    /COST, CLIQUES, RLT, RLTCOMP/
        VERTRLT1    /COST, CLIQUES, RLT, CLIQUERLT/;
OPTION LIMROW = 0, LIMCOL = 0;
SOLVE VERTLP USING RMIP MAXIMIZING Z;
        VLP = Z.L;
        FRAC = SUM(J, 1$(X.L(J) GT 0 AND X.L(J) LT 1));
SOLVE VERTIP USING MIP MAXIMIZING Z;
        VIP = Z.L;
        RELGAP = ( (VLP - VIP) / VIP ) * 100;
        ACTGAP = VLP - VIP;
        BIN = SUM(J, X.L(J));
DISPLAY VLP, VIP, RELGAP, ACTGAP, FRAC, BIN;
SOLVE VERTRLT0 USING RMIP MAXIMIZING Z;
        BIN = SUM(J, 1$(X.L(J) EQ 1));
        FRAC = SUM(J, 1$(X.L(J) GT 0 AND X.L(J) LT 1));
DISPLAY BIN, FRAC;
SOLVE VERTRLT1 USING RMIP MAXIMIZING Z;
        BIN = SUM(J, 1$(X.L(J) EQ 1));
DISPLAY BIN;
```

8.2. GAMS code for random ATSP problems.

```
$TITLE  RLT1 tightened MTZ formulation
$OFFUPPER
$OFFSYMXREF OFFSYMLIST OFFUELLIST OFFUELXREF
SET  I  variable index set          /1 * 50 /;
```

```
SET  I1(I)    index subset            /2 * 50 /;
ALIAS(I1,J1); ALIAS(I, J);
SCALARS   N          /50/
          rhs        /1/;
OPTION SEED= 4703;
OPTION RESLIM = 100000;
OPTION ITERLIM = 50000;
PARAMETER   C(I,I)    cost matrix for arcs;
            C(I,J)$(ORD(I) LT ORD(J)) = 1 + FLOOR( UNIFORM(0,99) );
            C(I,J)$(ORD(J) LT ORD(I)) = C(J,I);
* Set the main diagonal components to zero
            C(I,I) = 0;
PARAMETER   IJ(I,J)   mapping set used to show which arcs are present;
* If an arc is present for I not equal to J assign a one.
            IJ(I,J) = 1$(C(I,J)  GT  0);
VARIABLES
   X(I,J)   arc set
   Y(I,J)  new RLT1 variables after eliminating z
   U(J)    MTZ variables
   Z       total cost scalar;
BINARY VARIABLE X;
POSITIVE VARIABLE Y;
INTEGER VARIABLE U;
   U.LO(J) = 1;
   U.UP(J) = (N -1);
   U.FX('1') = 0;
EQUATIONS
   COST                 define objective function
   OUTGOING(I)          sum outgoing equals one
```

```
   INCOMING(J)         sum incoming equals one

   SD1(I)

   SD2(J)

   SD3(J)

   SD4(J)

   SD5(I,J)

   SD6(I,J)

   SD7(I,J)

   SD8(I,J);

   COST ..          Z =E= SUM((I,J), C(I,J) * X(I,J) );

   OUTGOING(I) ..   SUM( J$(ORD(I) NE ORD(J)),  X(I,J))  =E= rhs;

   INCOMING(J) ..   SUM( I$(ORD(I) NE ORD(J)),  X(I,J))  =E= rhs;

* RLT1 constraints.

   SD1(I)$(ORD(I) NE 1) .. ((N-1)*X(I,'1')) + SUM(J$(ORD(I) NE ORD(J)),
                                Y(I,J)$(ORD(J) NE 1)) =E= U(I);

   SD2(J)$(ORD(J) NE 1) .. SUM( I$(ORD(I) NE ORD(J)), Y(I,J)$(ORD(I) NE 1)) + 1
                                =E= U(J);

   SD3(J)$(ORD(J) NE 1) ..  U(J) - (1 - X('1',J)) - ((N-3)*X(J,'1')) =G= rhs;

   SD4(J)$(ORD(J) NE 1) ..  U(J) + ((N-3)*X('1',J)) - X(J,'1') =L= (N - 2);

   SD5(I,J)$(ORD(I) NE ORD(J) AND ORD(I) NE 1 AND ORD(J) NE 1) .. Y(I,J) -
                                X(I,J) =G= 0;

   SD6(I,J)$(ORD(I) NE ORD(J) AND ORD(I) NE 1 AND ORD(J) NE 1) .. ((N -
                                2)*X(I,J)) - Y(I,J) =G= 0;

   SD7(I,J)$(ORD(I) NE ORD(J) AND ORD(I) NE 1 AND ORD(J) NE 1) .. Y(I,J) +
                                Y(J,I) - U(J) - ((N - 2)*X(I,J))
                                    + ((N - 1)*(1 - X(J,I))) =G= 0;

   SD8(I,J)$(ORD(I) NE ORD(J) AND ORD(I) NE 1 AND ORD(J) NE 1) .. U(J) + X(J,I)
                                - Y(I,J)  - Y(J,I) =G= 1;

MODEL   RLT1LP    /ALL/
```

```
        RLT1IP     /ALL/;
OPTION LIMROW = 0, LIMCOL = 0;
SOLVE RLT1LP USING RMIP MINIMIZING Z;
SOLVE RLT1IP USING MIP MINIMIZING Z;
```

The three page vita has been removed from the scanned document. Page 1 of 3

The three page vita has been removed from the scanned document. Page 2 of 3

The three page vita has been removed from the scanned document. Page 3 of 3