Tight Flow-Based Formulations for the Asymmetric Traveling Salesman Problem and Their Applications to some Scheduling Problems

Pei-Fang Tsai

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

Dr. Subhash C. Sarin, Chair

Dr. Ebru K. Bish

Dr. Deborah F. Cook

Dr. Hanif D. Sherali

May 09, 2006

Blacksburg, Virginia

Keywords: Asymmetric Traveling Salesman Problem (ATSP), Multiple-Asymmetric Traveling Salesman Problem (mATSP), Reformulation-Linearization Technique (RLT), Precedence Constraints, Hot Strip Rolling Scheduling Problem

© 2006, Pei-Fang Tsai

Tight Flow-Based Formulations for the Asymmetric Traveling Salesman Problem and Their Applications to some Scheduling Problems

Pei-Fang Tsai

(Abstract)

This dissertation is devoted to the development of new flow-based formulations for the asymmetric traveling salesman problem (ATSP) and to the demonstration of their applicability in effectively solving some scheduling problems. The ATSP is commonly encountered in the areas of manufacturing planning and scheduling, and transportation logistics. The integration of decisions pertaining to production and shipping, in the supply chain context, has given rise to an additional and practical relevance to this problem especially in situations involving sequence-dependent setups and routing of vehicles. Our objective is to develop new ATSP formulations so that algorithms can be built by taking advantage of their relaxations (of integer variables, thereby, resulting in linear programs) to effectively solve large-size problems.

In view of our objective, it is essential to have a formulation that is amenable to the development of an effective solution procedure for the underlying problem. One characteristic of a formulation that is helpful in this regard is its tightness. The tightness of a formulation usually refers to the quality of its approximation to the convex hull of integer feasible solutions. Another characteristic is its compactness. The compactness of a formulation is measured by the number of variables and constraints that are used to formulate a given problem. Our formulations for the ATSP and the scheduling problems that we address are both tight and compact.

We present a new class of polynomial length formulations for the asymmetric traveling salesman problem (ATSP) by lifting an ordered path-based model using logical restrictions in concert with the Reformulation-Linearization Technique (RLT). We show that a relaxed version of this formulation is equivalent to a flow-based ATSP model, which, in turn, is tighter than the formulation based on the exponential number of

Dantzig-Fulkerson-Johnson (DFJ) subtour elimination constraints. The proposed lifting idea is applied to derive a variety of new formulations for the ATSP, and a detailed analysis of these formulations is carried out to show that some of these formulations are the tightest among those presented in the literature. Computational results are presented to exhibit the relative tightness of our formulations and the efficacy of the proposed lifting process.

While the computational results demonstrate the efficacy of employing the proposed theoretical RLT and logical lifting ideas, yet it remains of practical interest to take due advantage of the tightest formulations. The key requirement to accomplish this is the ability to solve the underlying LP relaxations more effectively. One approach, to that end, is to solve these LP relaxations to (near-) optimality by using deflected subgradient methods on Lagrangian dual formulations. We solve the LP relaxation of our tightest formulation, ATSP6, to (near-) optimality by using a deflected subgradient algorithm with average direction strategy (SA_ADS) (see Sherali and Ulular [69]). We also use two nondifferentiable optimization (NDO) methods, namely, the variable target value method (VTVM) presented by Sherali *et al*. [66] and the trust region target value method (TRTV) presented by Lim and Sherali [46], on the Lagrangian dual formulation of ATSP6. The preliminary results show that the near-optimal values obtained by the VTVM on solving the problem in the canonical format are the closest to the target optimal values. Another approach that we use is to derive a set of strong valid inequalities based on our tighter formulations through a suitable surrogation process for inclusion within the more compact manageable formulations. Our computational results show that, when the dual optimal solution is available, the associated strong valid inequalities generated from our procedure can successfully lift the LP relaxation of a less tight formulation, such as ATSP2R^{-}, to be as tight as the tightest formulation, such as ATSP6.

We extend our new formulations to include precedence constraints in order to enforce a partial order on the sequence of cities to be visited in a tour. The presence of precedence constraints within the ATSP framework is encountered quite often in practice. Examples include: disassembly optimization (see Sarin *et al*. [62]), and scheduling of wafers/ ICs on automated testing equipments in a semiconductor manufacturing facility

(see Chen and Hsia [17]); among others. Our flow-based ATSP formulation can very conveniently capture these precedence constraints. We also present computational results to depict the tightness of our precedence-constrained asymmetric traveling salesman problem (PCATSP) formulations.

We, then, apply our formulations to the hot strip rolling scheduling problem, which involves the processing of hot steel slabs, in a pre-specified precedence order, on one or more rollers. The single-roller hot strip rolling scheduling problem can be directly formulated as a PCATSP. We also consider the multiple-roller hot strip rolling scheduling problem. This gives rise to the multiple-asymmetric traveling salesman problem (mATSP). Not many formulations have been presented in the literature for the mATSP, and there are none for the mATSP formulations involving a precedence order among the cities to be visited by the salesmen, which is the case for the multiple-roller hot strip rolling scheduling problem. To begin with, we develop new formulations for the mATSP and show the validity of our formulations, and present computational results to depict their tightness. Then, we extend these mATSP formulations to include a pre-specified, special type of precedence order in which to process the slabs, and designate the resulting formulations as the restricted precedence-constrained multiple-asymmetric traveling salesman problem (rPCmATSP) formulations. We directly formulate the multiple-roller hot strip rolling scheduling problem as a rPCmATSP. Furthermore, we consider the hot strip rolling scheduling problem with slab selection in which not all slabs need to be processed. We model the single-roller hot strip rolling scheduling problem with slab selection as a multiple-asymmetric traveling salesman problem with exactly two traveling salesmen. Similarly, the multiple-roller hot strip rolling scheduling problem with slab selection is modeled as a multiple-asymmetric traveling salesman problem with (m+1) traveling salesmen.

A series of computational experiments are conducted to exhibit the effectiveness of our formulations for the solution of hot strip rolling scheduling problems. Furthermore, we develop two mixed-integer programming algorithms to solve our formulations. These are based on Benders´ decomposition [13] and are designated Benders´ decomposition and Modified Benders´ methods. In concert with a special type of precedence order presented in the hot strip rolling scheduling problems, we further introduce an adjustable

density ratio of the associated precedence network and we use randomly generated test problems to study the effect of various density ratios in solving these scheduling problems. Our experimentation shows the efficacy of our methods over CPLEX.

Finally, we present a compact formulation for the job shop scheduling problem, designated as JSCD (job shop conjunctive-disjunctive) formulation, which is an extension of our ATSP formulations. We use two test problems given in Muth and Thompson [53] to demonstrate the optimal schedule and the lower bound values obtained by solving the LP relaxations of our formulations. However, we observe that the lower bound values obtained by solving the LP relaxations of all variations of our JSCD formulation equal to the maximum total processing time among the jobs in the problem.

Acknowledgements

First and foremost, I would like to thank my advisor, Dr. Subhash C. Sarin, for his guidance and generous support throughout my years at Virginia Tech. I have learnt so much from working with him and his patience has endured my obstinate anxiety during the countless hours in research meetings. I specially appreciate Dr. Hanif D. Sherali, from whom I have learnt a lot about theories and techniques in optimization, for his enlightened suggestion and magical touch on this research. I also want to give thanks to my committee members, Dr. Ebru Bish and Dr. Deborah Cook, for their kindness and encouragement during our discussion. I thank Dr. Barbara Fracticelli for generously helping me adopting the culture difference and for meeting her vivid family members, Mr. Tom Fracticelli, Cecilia, and Josephine.

I want to send my grateful heart to Ms. Lovedia Cole for her excellent assistance thought all administrative processes along the way and for comforting me like a family. A special thank you goes to Dr. Frank Chen for his attentive care since my first day in Blacksburg and to Dr. Benjamin Blanchard for his charismatic guidance. I would like to thank Dr. Andrew Wei-Hua Wang at Tunghai University for his confidence in me and for recommending me to go for PhD studies. I thank Dr. Kimberly Ellis, Dr. Joel Nachlas, and Dr. John Shewchuk for their trust during my work with them as a teaching assistant, and a special thanks to Dr. Patrick Koelling for lending me his patient ears.

I would not be able to commence my journey here and continue without the great supports from all friends around me. Thanks to my officemates and good friends Gregory Beskow, Amy Brown, Dr. Michael McCrea, Dr. Pornthipa (Morr) Ongkunaruk, Dr. Churlzu Lim, Dr. Xiaomei Zhu, Qiong Wang, Ming Chen, and all fellow colleagues for making this journey bearable. I would like to thank my dear friends Yi-Chun (Popo) Tsai, Suh-Liang Ou, Karen Hsu, Cathy Hsieh, and Tseng-Wei Chung for their encouragement. I also thank my CSA friends Shun-Hua Li, Claire Chang, Anney Yeh, Alice Kuo, Lianna Su and Steven Tsai, Shu-Yi Tsai and Hung-Da Wan, Eric Chia, Bert Chen, Renzo Chen, Ball Lin, Truman Chang, and many CSA members for making my life in Blacksburg a lot of fun.

My final thank you is for my dearest parents, Sho-Chin Tsai-Fang and Li-Chao Tsai, who might know nothing about my research but understand me and always be there for me no matter what. I would not even be able to come this far without their support, emotionally and financially. They are the reason that keeps me moving on and trying to achieve anything that could make them proud. I would like to dedicate this dissertation to my parents, as an infinitesimal return for their unconditional love.

Table of Content

List of Tables

List of Figures

Chapter 1
Introduction

The literature in the area of production scheduling is replete with instances where a scheduling problem is solved with some type of a dispatching rule. The main reason for this, of course, is the complexity of the scheduling problems involved, which in general, are NP-hard. The combinational nature of the scheduling problems, inherently, leads to the investigation of a branch and bound type of procedure for their solution. There is, however, a dearth of approaches for these problems that rely on their tight mathematical programming formulations to provide good lower bounds for use in a branch and bound procedure. This work is motivated by a desire to develop such tight formulations for scheduling problems involving sequence-dependent setup times.

In a practical production scheduling environment, it is not uncommon to find instances that require sequence-dependent setup times (SDST) for the processing of the jobs (work) on machines (resources) [59]. In a survey of industrial managers conducted by Panwalkar *et al*. [56], it is found that approximately 15% of the managers surveyed encounter sequence-dependent setup times in all of their operations, while about three quarters of the managers surveyed encounter at least some operations that require SDST. Wilbrecht and Prescott [74] report that SDST is significant when a job shop runs at or near full capacity. All of these underlie the importance of considering SDST in its own right and not just combining it with the job processing times.

The SDST problem that we address in this research involves minimization of the makespan. All the jobs are assumed to be available for processing at time zero. The scheduling environment may consist of one machine or multiple, parallel machines. The single-machine SDST problem for the objective of minimizing the makespan is equivalent to the asymmetric traveling salesman problem (ATSP) (see Baker [5]). Therefore, the first topic of our investigation in this research is the ATSP. We, then,

expand our investigation along two directions. The first expansion involves inclusion of precedence constraints within the ATSP framework. This alludes to the consideration of the ATSP over a network. The second expansion involves the application of our new ATSP formulations to two classes of commonly encountered problems. The first of these problems pertains to the hot strip rolling scheduling problem. This is an interesting application, because it leads to the development of new formulations for the multiple-asymmetric traveling salesman problem (mATSP), albeit including sequence-dependent setup times and a special type of precedence among the jobs that are processed on hot rollers. The second application of our new formulations that we present is the job shop scheduling problem, which is an important problem in its own right.

The ATSP has been discussed extensively in the literature. This problem involves the determination of a tour, which starting from a base city visits all other given cities one and only once and terminates at the base city while minimizing the total distance traveled. Various formulations for this problem differ in the way the subtour elimination constraints (SEC) are represented. The first of these formulations was presented by Dantzig, Fulkerson, and Johnson [19], which involves SECs that grow exponentially in the number of cities that are to be visited. Compact formulations for the problem, which involve SECs that grow polynomially in the number of cities to be visited, have also been proposed (see Miller *et al.* [51], Desrochers and Laporte [20], Sherali and Driscoll [67], Gouveia and Pires [26], and Sarin *et al.* [61]). Moreover, Sarin *et al.* [61] have shown that a lifted version of their formulation gives tighter lower bounds than those given by Miller *et al.* [51], Desrochers and Laporte [20], Sherali and Driscoll [67], and Gouveia and Pires [26]. On the other hand, Wong [75] has presented a flow-based formulation, which is compact and equivalent to the formulation in Dantzig *et al.* [19]. Here, we present new flow-based formulation for the asymmetric traveling salesman problem that further strengthens the formulation given in Sarin *et al.* [61]. We use the Reformulation-Linearization Technique (RLT) (see Sherali and Adams [63,64,65]) to derive specialized inequalities. The validity of these inequalities is shown. Also, we show the dominance of our formulation over other ATSP formulations. Computational results are presented to further exhibit this dominance of our formulations.

The presence of precedence constraints within the ATSP framework is encountered quite often in practice. Examples include: disassembly optimization (see Sarin *et al.* [62]), and scheduling of wafers/ ICs on automated testing equipments in a semiconductor manufacturing facility (see Chen and Hsia [17]); among others. Our flow-based ATSP formulation can very conveniently capture these precedence constraints. We also present computational results to depict the tightness of our precedence-constrained ATSP (PCATSP) formulation.

The hot strip rolling scheduling problem that we address involves the processing of hot steel slabs, in a pre-specified precedence order, on one or more rollers. The single-roller hot strip rolling scheduling problem is similar to the PCATSP, except for the special type of precedence order in which to process the slabs. The multiple-asymmetric traveling salesman problem may arise in its own right in the presence of multiple rollers. Not many formulations have been presented in the literature for the mATSP, and there are none for the mATSP formulations involving a precedence order among the cities to be visited by the salesmen, which is the case of the multiple-roller hot strip rolling scheduling problem. We develop new formulations for the mATSP with and without the inclusion of pre-specified precedence order among the cities (slabs). The special type of precedence order in which the slabs are processed permits the ease of its inclusion in our restricted precedence-constrained mATSP (rPCmATSP) formulation. These formulations are also used to model the hot strip rolling scheduling problem with slab selection, for the case of single roller and multiple rollers. We show the validity of our formulations and present computational results to depict their tightness.

The remainder of this dissertation is organized as follows. In Chapter 2, we review the relevant formulations presented in the literature for the asymmetric traveling salesman problem (ATSP) and the multiple-asymmetric traveling salesman problem (mATSP).

In Chapter 3, we present a new class of polynomial length formulations for the ATSP with and without precedence constraints by lifting an ordered path-based model using logical restrictions in concert with the Reformulation-Linearization Technique (RLT). We start with a new Hamiltonian path-based formulation, designated as ATSP0, which is

enhanced by certain logical constraints in Section 3.1. We first establish the validity of this formulation, and later on, we tighten the polytope of this formulation by applying the first-order Reformulation-Linearization Technique (RLT) of Sherali and Adams [63,64,65]. This gives rise to three RLT-lifted constraints. We augment our ATSP0 model with these three RLT-lifted constraints to come up with ATSP1 model. Moreover, we try several combinations of available constraint sets with these RLT-lifted constraints and present a class of new polynomial length formulations in Section 3.2.

In Section 3.3, we compare our new formulations with the formulation based on the exponential number of Dantzig-Fulkerson-Johnson (DFJ) subtour elimination constraints (designated as ATSP-FL). We show that our new formulations, those containing the RLT-lifted constraints, afford a tighter LP relaxation than ATSP-FL. We further show the dominance of our new formulations over those that are presented in the literature for the ATSP. In our computational results, presented in Section 3.4, this dominance is reflected by the tightness of the lower bound values obtained by solving the LP relaxations of our formulations.

In Section 3.5, we consider the Lagrangian dual formulation of the LP relaxations of ATSP formulations and solve this Lagrangian dual by using a subgradient algorithm with average direction strategy (SA_ADS) [69]. We also use two nondifferentiable (NDO) optimization algorithms, namely, the variable target value method (VTVM) presented by Sherali *et al*. [66] and the trust region target value (TRTV) method presented by Lim and Sherali [46]. Another alternative that we propose in Section 3.6 is a surrogation process to derive a set of strong valid inequalities based on tighter formulations, and use these cuts within more compact manageable formulations.

Finally, we extend our new formulations for the ATSP to the precedence-constrained asymmetric traveling salesman problem (PCATSP) in Section 3.7. By the nature of our new formulations, it is straightforward to enforce the precedence constraints without violating the validity of the underlying ATSP formulations. Hence, we show that the dominance relations for the ATSP formulations in Section 3.3 are applicable to the formulations for the PCATSP as well. Computational experiments are conducted on the PCATSP in Section 3.8, and the results reflect the dominance of our formulations

because of the largest lower bound values obtained (by solving their LP relaxations) by our formulations.

In Chapter 4, we apply and extend our ATSP formulations to various hot strip rolling scheduling problems. In Section 4.1, we describe the hot strip rolling scheduling problem and address its characteristics. We review the formulations and algorithms that are presented for this problem in the literature, in Section 4.2. In Section 4.3, we first present four hot strip rolling scheduling problems. In Section 4.3.1, we consider the single-roller hot strip rolling scheduling problem and model it as a precedence-constrained ATSP (PCATSP). When the schedules for multiple rollers have to be decided simultaneously, we formulate the underlying scheduling problem as a multiple-asymmetric traveling salesman problem (mATSP) with restricted precedence constraints. We present new polynomial length formulations for the mATSP in Section 4.3.2. Then, we extend our mATSP formulations to the multiple-roller hot strip rolling scheduling problem involving the restricted type of precedence constraints encountered during the processing of the slabs in Section 4.3.2.2. We have modified these formulations to model the hot strip rolling scheduling problem with slab selection, for the case of single roller and multiple rollers, in Section 4.3.2.3 and in Section 4.3.2.4, respectively. In Section 4.4, we design a series of experiments and present our computational results by solving our formulations using Benders´ decomposition [13] and Modified Benders´ methods, and CPLEX.

In Chapter 5, we consider the job shop scheduling problem. In Section 5.1, we review the formulations available in the literature. Then, we present our compact formulation for the job shop scheduling problems in Section 5.2. Some numerical results, on the use of this formulation, are presented in Section 5.3.

Chapter 2

Literature Review

This chapter briefly reviews the literature on the ATSP and mATSP problems. We begin in Section 2.1 by reviewing several formulations for the asymmetric traveling salesman problem (ATSP). Section 2.2 provides formulations for the multiple-asymmetric traveling salesman problem (mATSP).

2.1 Literature on Asymmetric Traveling Salesman Problem

The traveling salesman problem (TSP) is to determine a tour which, starting from a base city, visits all other given cities one and only once and terminates at the base city while minimizing the total distance traveled. Let c_{ij} be the distance from city i to city j. If $c_{ij} = c_{ji}$ for all i and j, then we designate the traveling salesman problem as a symmetric traveling salesman problem (STSP); otherwise, it is called an asymmetric traveling salesman problem (ATSP).

A key component of the ATSP formulations is the subtour elimination constraints (SECs). Several different formulations for the SECs have been presented in the literature. The first of these was proposed by Dantzig, Fulkerson, and Johnson [19], and it is also known in the literature as the DFJ constraints.

ATSP-DFJ: Minimize $\sum_{i=1}^{n} \sum_{j=1, j\neq i}^{n} c_{ij} x_{ij}$ (2.1a)

subject to

$$\sum_{i=1, i\neq j}^{n} x_{ij} = 1 \ , \ \forall j = 1,...,n \quad (2.1b)$$

$$\sum_{j=1, j\neq i}^{n} x_{ij} = 1 \ , \ \forall i = 1,...,n \quad (2.1c)$$

$$\sum_{i\in S}\sum_{j\in S} x_{ij} \le |S|-1,\ \forall S \subseteq \{2,...,n\}\ ,\ |S| \ge 2 \qquad (2.1d)$$

$$x_{ij} = \{0,1\}\ ,\ \forall i, j = 1,...,n\,. \qquad (2.1e)$$

Note that the variable $x_{ij} = 1$ if node i immediately precedes node j on a tour, and 0, otherwise; and n is the total number of cities. The subtour elimination constraints given by (2.1d) are known to be facet defining for the polytope of the ATSP (see Grotschel and Padberg [29]). However, one of the difficulties in implementing ATSP-DFJ is the exponential number of SECs that it contains. To overcome this, Grotschel and Padberg [28] and Carr and Lancia [15] have suggested some implementation strategies to reduce the size of ATSP-DFJ.

Several compact formulations for the ATSP have also been presented in the literature. Wong [75] argues that any feasible solution of the LP relaxation of ATSP-DFJ is the same as enforcing the maximum flow of at least a unit from the source node to all other nodes. Consequently, he proposes a compact formulation for the ATSP that utilizes the maximum flow constraints. We designate this formulation as ATSP-FL. It is given as follows:

ATSP-FL: Minimize $\sum_{i=1}^{n}\sum_{j=1,\,j\neq i}^{n} c_{ij}x_{ij}$ (2.2a)

subject to

(2.1b) and (2.1c) from ATSP-DFJ

$$y_{ij}^{u} \le x_{ij}\ ,\ \forall u = 2,...,n\ ,\ \forall i, j = 1,...,n\ ,\ i \neq j \qquad (2.2d)$$

$$\sum_{j=2}^{n} y_{1j}^{u} - \sum_{j=2}^{n} y_{j1}^{u} = 1\ ,\ \forall u = 2,...,n \qquad (2.2e)$$

$$\sum_{j=1,\,j\neq i}^{n} y_{ij}^{u} - \sum_{j=1,\,j\neq i}^{n} y_{ji}^{u} = 0\ ,\ \forall u = 2,...,n\ ,\ \forall i = 2,...,n\ ,\ u \neq i \qquad (2.2f)$$

$$\sum_{j=1,\,j\neq u}^{n} y_{uj}^{u} - \sum_{j=1,\,j\neq u}^{n} y_{ju}^{u} = -1\ ,\ \forall u = 2,...,n \qquad (2.2g)$$

$$x_{ij} = \{0,1\}\ ,\ \forall i, j = 1,...,n\,;\ y_{ij}^{u} \ge 0\ ,\ \forall i, j = 1,...,n\ ,\ \forall u = 2,...,n\,. \qquad (2.2h)$$

Note that the variable y_{ij}^{u} represents the flow from node 1 to node u via traveling on the arc (i, j). The constraint sets (2.2e,g) indicate that the maximum out-flow from node 1 and the inflow to any node u, other than node 1, must be 1. Constraint set (2.2f) captures flow balance at all intermediate nodes, that is, for nodes other than node 1 and u.

Another compact formulation is due to Miller-Tucker-Zemlin (MTZ) [51] in which the subtour elimination constraints (2.1d) are replaced as follows. Let u_i be an arbitrary real number for some node i, $\forall i = 1,...,n$. Without loss of generality, we can assume $u_1 = 0$ and $1 \le u_i \le (n-1)$, $\forall i = 2,...,n$. Their formulation is as follows.

ATSP-MTZ: Minimize $\sum_{i=1}^{n} \sum_{j=1, j\neq i}^{n} c_{ij} x_{ij}$ (2.3a)

subject to

(2.1b) and (2.1c) from ATSP-DFJ

$$u_i - u_j + n \cdot x_{ij} \le (n-1) \ , \ \forall i, j = 2,...,n \ , \ i \neq j \tag{2.3d}$$

$$x_{ij} = \{0,1\} \ , \ \forall i, j = 1,...,n \, ; \, 1 \le u_i \le (n-1) \, , \ \forall i = 2,...,n \text{ and } u_1 = 0 \, . \tag{2.3e}$$

Although the ATSP-MTZ formulation is compact and contains only $(n-1)^2$ subtour elimination constraints, it does not result in as tight an LP relaxation as ATSP-DFJ does. Desrochers and Laporte [20] propose tighter ATSP-MTZ constraints (designated ATSP-DL) by using the sequential lifting technique. The formulation by Sherali and Driscoll [67] (designated ATSP-SD) is obtained by lifting ATSP-MTZ constraints using a special version of the Reformulation-Linearization technique (RLT) of Sherali and Adams [63,64,65]. We present the ATSP-SD below.

ATSP-SD Minimize $\sum_{i=1}^{n} \sum_{j=1, j\neq i}^{n} c_{ij} x_{ij}$ (2.4a)

subject to

(2.1b) and (2.1c) from ATSP-DFJ

$$\sum_{j=2, j\neq i}^{n} y_{ij} + (n-1) \cdot x_{i1} = u_i \ , \ \forall i = 2,...,n \tag{2.4d}$$

$$\sum_{i=2, i\neq j}^{n} y_{ij} + 1 = u_j \ , \ \forall j = 2,...,n \tag{2.4e}$$

$$x_{ij} \le y_{ij} \le (n-2)\cdot x_{ij} \ , \ \forall i, j = 2,...,n \ , \ i \neq j \tag{2.4f}$$

$$u_j + (n-2)\cdot x_{ij} - (n-1)\cdot\left(1 - x_{ji}\right) \le y_{ij} + y_{ji} \le u_j - \left(1 - x_{ji}\right) \ , \ \forall i, j = 2,...,n \ , \ i \neq j \tag{2.4g}$$

$$1 + \left(1 - x_{1j}\right) + (n-3)\cdot x_{j1} \le u_j \le (n-1) - (n-3)\cdot x_{1j} - \left(1 - x_{j1}\right) \ , \ \forall j = 2,...,n \tag{2.4h}$$

$$x_{ij} = \{0,1\} \ , \ \forall i, j = 1,...,n \ ; \ y_{ij} \ge 0 \ , \ \forall i, j = 1,...,n \tag{2.4i}$$

$$1 \le u_i \le (n-1) \ , \ \forall i = 2,...,n \text{ and } u_1 = 0 \ . \tag{2.4j}$$

Gouveia and Pires [26] have proposed a tightened MTZ formulation by disaggregating the MTZ constraints (designated DMTZ formulation). Sarin, Sherali, and Bhootra [61] further tighten this formulation by defining an auxiliary variable y_{ij}, which represents whether node i precedes node j. Their formulation is as follows.

ATSP-SSB Minimize $\sum_{i=1}^{n} \sum_{j=1, j\neq i}^{n} c_{ij} x_{ij}$ (2.5a)

subject to

(2.1b) and (2.1c) from ATSP-DFJ

$$y_{ij} \ge x_{ij} \ , \ \forall i, j = 2,...,n \ , \ i \neq j \tag{2.5d}$$

$$y_{ij} + y_{ji} = 1 \ , \ \forall i, j = 2,...,n \ , \ i \neq j \tag{2.5e}$$

$$y_{ij} + y_{jk} + y_{ki} \le 2 \ , \ \forall i, j, k = 2,...,n \ , \ i \neq j \neq k \tag{2.5f}$$

$$x_{1j} + x_{j1} \le 1 \ , \ \forall j = 2,...,n \tag{2.5g}$$

$$x_{ij} = \{0,1\} \ , \ \forall i, j = 1,...,n \ ; \ y_{ij} \ge 0 \ , \ \forall i, j = 1,...,n \ . \tag{2.5h}$$

This ATSP-SSB formulation uses the simple fact that two nodes, node i and node j, can only have either node i precede node j or node j precede node i, but not both (constraint set (2.5e)). Constraint set (2.5d) forces node i to precede node j if node i 'immediately' precedes node j. Constraint set (2.5f) represents the three-nodes DFJ

subtour elimination constraints. Constraint set (2.5g) is used to tighten the LP relaxation of the formulation. Sarin *et al.* [61] have shown that this formulation is valid for the ATSP. They further tightened their formulation by sequentially lifting (2.5f), and the lifted ATSP-SSB formulations are shown to yield tighter z_{LP} values than those obtained by ATSP-DL (Desrochers and Laporte [20]), ATSP-SD (Sherali and Driscoll [67]), and DMTZ (Gouveia and Pires [26,27]) (see Sarin *et al.* [61]).

2.2 Literature on Multiple-Asymmetric Traveling Salesman Problem

The multiple-asymmetric traveling salesman problem (mATSP) is a generalization of the single-asymmetric traveling salesman problem (ATSP) and can be defined as follows: Given a set of cities and m traveling salesmen, determine m tours, one for each salesman, such that, starting from the same base city, each salesman visits a subset of cities and returns to the base city; each city is to be visited by only one salesman, and the objective is to minimize the total distance traveled by all salesmen.

Several integer programming formulations have been proposed for the mATSP in the literature. Laporte and Nobert [41] have extended the ATSP formulation of Dantzig-Fulkerson-Johnson (DFJ) to the m-salesman case and have applied a linear programming based method for its solution, by using subtour elimination and integrality constraints as cutting planes. Gavish and Srikanth [24] and Ali and Kennington [1] have used the same formulation as in Laporte and Nobert [41] and have proposed a branch-and-bound procedure with a lower bound estimated by solving a Lagrangian relaxation of a degree-constrained minimal spanning tree (see Held and Karp [30,31]).

Svestka and Huckfeldt [70], Gavish [23], Christofides *et al.* [18], and Bektas [12] have extended the Miller-Tucker-Zemlin (MTZ) formulation of the single-asymmetric traveling salesman problem to the m-salesman case. Moreover, Svestka and Huckfeldt [70] have developed a branch and bound algorithm where a subtour is avoided by assigning an infinite distance on one arc of that subtour during the branching procedure. The extended MTZ formulation (designated as mATSP-MTZ) can be presented as follows.

Let u_i be an arbitrary real number for some node i, $\forall i = 1,...,n$. Without loss of generality, we can assume $u_1 = 0$ and $1 \le u_i \le p$, $\forall i = 2,...,n$, where p denotes the maximum number of cities that can be visited by any salesman [51]. We have,

mATSP_MTZ : Minimize $\sum_{i=1}^{n} \sum_{j=1, j\neq i}^{n} c_{ij} x_{ij}$ (2.6a)

subject to

$$\sum_{j=2, j\neq i}^{n} x_{1j} = m \quad (2.6b)$$

$$\sum_{i=2, i\neq j}^{n} x_{i1} = m \quad (2.6c)$$

$$\sum_{i=1, i\neq j}^{n} x_{ij} = 1 \ , \ \forall j = 2,...,n \quad (2.6d)$$

$$\sum_{j=1, j\neq i}^{n} x_{ij} = 1 \ , \ \forall i = 2,...,n \quad (2.6e)$$

$$u_i - u_j + p \cdot x_{ij} \le (p-1), \ \forall i, j = 2,...,n \ , \ i \neq j \quad (2.6f)$$

$$x_{ij} \in \{0,1\} \ , \ \forall i, j = 1,...,n \ , \ i \neq j \ ; \ u_i \ge 0 \ , \ \forall i = 2,...,n \ . \quad (2.6g)$$

Note that the variable $x_{ij} = 1$ if node i immediately precedes node j on a tour, and 0, otherwise; and n is the number of cities. The constraint sets (2.6d,e) are the assignment constraints. The constraint sets (2.6b,c) ensure that exactly m salesmen depart from and return to the base city. The constraint set (2.6f) is the Miller-Tucker-Zemlin (MTZ) subtour elimination constraint.

Bektas [12] has tightened this MTZ formulation (designated here as mATSP_Bektas) with an additional constraint that imposes a minimum number (K) of cities to be included in a tour. We present Bektas' formulation next.

mATSP_Bektas: Minimize $\sum_{i=1}^{n} \sum_{j=1, j\neq i}^{n} c_{ij} x_{ij}$ (2.7a)

subject to

$$\sum_{j=2,\, j\neq i}^{n} x_{1j} = m \tag{2.7b}$$

$$\sum_{i=2,\, i\neq j}^{n} x_{i1} = m \tag{2.7c}$$

$$\sum_{i=1,\, i\neq j}^{n} x_{ij} = 1 \ , \ \forall j = 2,...,n \tag{2.7d}$$

$$\sum_{j=1,\, j\neq i}^{n} x_{ij} = 1 \ , \ \forall i = 2,...,n \tag{2.7e}$$

$$u_i + (L-2)\cdot x_{1i} - x_{i1} \leq (L-1), \ \forall i = 2,...,n \tag{2.7f}$$

$$u_i + x_{1i} + (2-K)\cdot x_{i1} \geq 2, \ \forall i = 2,...,n \tag{2.7g}$$

$$u_i - u_j + L\cdot x_{ij} + (L-2)\cdot x_{ji} \leq (L-1), \ \forall i, j = 2,...,n, \ i \neq j \tag{2.7h}$$

$$x_{ij} \in \{0,1\} \ , \ \forall i, j = 1,...,n, \ i \neq j; \ u_i \geq 0, \ \forall i = 2,...,n. \tag{2.7i}$$

The constraint sets (2.7b,c) ensure that exactly m salesmen depart from and return to the base city. The constraint sets (2.7d,e) are the assignment constraints. The constraint sets (2.7f,g) are used to impose bounds on the number of cities that a salesman can visit, where L denotes the maximum number of cities that can be visited by any salesman. As noted earlier, we can assume $u_1 = 0$ and $1 \leq u_i \leq L$, $\forall i = 2,...,n$. Finally, the constraint set (2.7h) constitutes the lifted Miller-Tucker-Zemlin (MTZ) subtour elimination constraint.

Another approach that has been presented in the literature to solve the mATSP (mTSP) is to transform this problem to an equivalent ATSP (TSP). Svestka and Huckfeldt [70] have proposed an augmented distance matrix by including additional $m-1$ rows and columns, where each new row and column is a duplicate of the first row and column of the original matrix and infinite distance is assigned to all new elements, assuming that city 1 is the base city. Jonker and Volgenant [39] suggest removing $m(m-2)$ duplicated edges between base cities and other cities in order to reduce degeneracy when more than 2 salesmen are involved. In the computational results, they show that their branch and bound procedure requires less computational effort when compared with that of Gavish and Srikanth [24], in which the mTSP is solved directly

using a branch and bound algorithm. However, the edge removing procedure is subject to heuristic rules, and hence, the optimality of their solution can not be guaranteed.

Bellmore and Hong [11], Hong and Padberg [33], and Rao [60] have presented equivalent single traveling salesman formulation for the mTSP when fixed charges are involved; i.e, an additional fixed charge f_i incurs when the traveling salesman i finishes his tour and this charge is independent of the length of his tour. Bellmore and Hong [11] propose a transformation of this mATSP to a standard ATSP containing $n+m-1$ nodes. However, Hong and Padberg [33] argue that, in the symmetric cases, the mTSP with fixed charge should be transformed to a standard TSP with $n+m+4$ nodes. Figure 2-1 represents a TSP with n cities and 5 salesmen [33]. For those edges that are not shown in G^* among nodes $\{n,\ldots,(n+m+4)\}$, the associated costs are infinite. Then, the fixed cost of each traveling salesman is deployed as follows. The costs of edges connecting city n or city $n+m$ with original cities $\{1,2,...,n-1\}$ are increased by $\frac{1}{2}\ f_1$, and the costs for the edges connecting cities $(n+i-1)$ with cities $\{1,2,...,n-1\}$ are increased by $\frac{1}{2}\ f_i$, $\forall i = 2,3,..,m$, accordingly. The costs for the remaining edges are zero. Rao [60] shows that it is only possible to transform the mTSP with fixed charge into a standard TSP with $n+m-1$ nodes when the number of traveling salesman is at most 2.

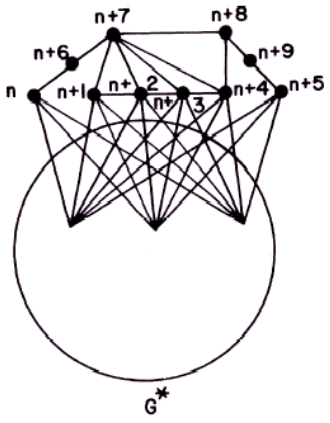

Figure 2-1 A TSP representation for the mTSP with fixed charges for m=5 (see Hong and Padberg [33])

Chapter 3
Formulations for the Asymmetric Traveling Salesman Problem with and without Precedence Constraints

In this chapter, we present a class of new polynomial length formulations for the asymmetric traveling salesman problem (ATSP) and prove that it generates a tighter LP relaxation than that generated by the formulation based on the exponential number of Dantzig-Fulkerson-Johnson (DFJ) subtour elimination constraints. Then, we extend our formulation to include precedence constraints in order to enforce a partial order on the sequence of cities to be visited in a tour. Computational results are also presented to exhibit the relative tightness of our formulations.

The organization of this chapter is as follows. In Section 3.1, we introduce a new Hamiltonian path-based formulation enhanced by some logical constraints. Further, we use the Reformulation-Linearization Technique (RLT) (see Sherali and Adams [63,64,65]) to derive specialized valid inequalities and provide a class of ATSP formulations in Section 3.2. In Section 3.3, we establish the dominance relationship between these formulations and present some computational results in Section 3.4. In Sections 3.5 and 3.6, we attempt to improve the efficiency of obtaining lower bounds (z_{LP}) and the effectiveness of obtaining integer solutions, respectively, of some of these formulations. Then, we extend these formulations to include precedence constraints in Section 3.7. This is followed by the computational results for this class of problems in Section 3.8.

3.1 Formulation ATSP0

We introduce a formulation of ATSP, which is based on variable definitions as used in Gouveia and Pires [26,27] and Sarin, Sherali, and Bhootra [61]. These variables are as follows. Let

$$x_{ij} = \begin{cases} 1, \text{ if city } i \text{ directly precedes city } j, \\ 0, \text{ otherwise, } \forall i, j = 1,...,n,\ i \neq j, \end{cases}$$

and

$$y_{ij} = \begin{cases} 1, \text{ if city } i \text{ precedes city } j \text{ (not necessarily directly),} \\ 0, \text{ otherwise, } \forall i, j = 2,...,n,\ i \neq j. \end{cases}$$

The $\boldsymbol{x}$-variables are the same as those used in ATSP-FL and define a Hamiltonian circuit, whereas for the $\boldsymbol{y}$-variables, we assume that city 1 is the base city so that these variables view the ATSP solution as a Hamiltonian path that commences at city 1. Hence, the indices for the $\boldsymbol{y}$-variables range over the cities $2,...,n$.

ATSP0: Minimize $\sum_{i=1}^{n} \sum_{j=1,\, j\neq i}^{n} c_{ij} x_{ij}$ (3.1a)

subject to

$$\sum_{v=1, v\neq i}^{n} x_{iv} = 1\ ,\ \forall i = 1,...,n \tag{3.1b}$$

$$\sum_{i=1, i\neq v}^{n} x_{iv} = 1\ ,\ \forall v = 1,...,n \tag{3.1c}$$

$$y_{ij} \geq x_{ij}\ ,\ \forall i, j = 2,...,n,\ i \neq j \tag{3.1d}$$

$$y_{ij} + y_{ji} = 1\ ,\ \forall i, j = 2,...,n,\ i \neq j \tag{3.1e}$$

$$y_{ij} \geq x_{1i}\ ,\ \forall i, j = 2,...,n,\ i \neq j \tag{3.1f}$$

$$y_{ji} \geq x_{i1}\ ,\ \forall i, j = 2,...,n,\ i \neq j \tag{3.1g}$$

$$-(1 - x_{iv}) \leq (y_{ij} - y_{vj}) \leq (1 - x_{iv}),\ \forall i, j, v = 2,...,n,\ i \neq j \neq v \tag{3.1h}$$

$$x_{ij} \in \{0,1\}, \forall i, j = 1,...,n,\ i \neq j,\ y_{ij} \geq 0,\ \forall i, j = 2,...,n,\ i \neq j. \tag{3.1i}$$

In ATSP0, (3.1b,c,i) are the standard assignment constraints; (3.1d,f,g) logically relate the precedence variables $\boldsymbol{y}$ to the immediate precedence variables $\boldsymbol{x}$; (3.1e) asserts that either i should precede j or vice versa, $\forall i, j = 2,...,n$, $i \neq j$, and (3.1h) enforces that if $x_{iv} = 1$ for any $i \neq v$, then for any other city j, we must have $y_{ij} = y_{vj}$ (whether this pair equals 0 or 1). Observe that due to (3.1e), this also then ensures that $y_{ji} = y_{jv}$. Formulation ATSP0 is similar to the model ATSP-SSB given in Sarin, Sherali, and Bhootra [61] (called ATSPxy therein), except for the additional logical relations (3.1f,g) used in ATSP0, and the use of the following set of three-city subtour elimination constraints in ATSP-SSB in lieu of (3.1h):

$$y_{ij} + y_{jk} + y_{ki} \leq 2, \ \forall i, j, k = 2,...,n, \ i \neq j \neq k. \tag{3.2}$$

Note that, by using (3.1e) to complement variables, the left-hand inequality in (3.1h) also reduces to the constraint $y_{ji} - y_{jv} \leq 1 - x_{iv}$, $\forall i, j, v = 2,...,n$, $i \neq j \neq v$, which was employed in Gouveia and Pires [27]. Nonetheless, for convenience in our analysis below, we begin by examining ATSP0 given by (3.1) above, and then consider further extensions and enhancements in the sequel.

Before proceeding further, we first establish the validity of the model ATSP0.

Proposition 3.1. The formulation ATSP0 given by (3.1) obviates any subtours, and hence, correctly models the asymmetric traveling salesman problem.

Proof. On the contrary, given any feasible solution $(\boldsymbol{x}, \boldsymbol{y})$ to ATSP0, suppose that the assignment solution $\boldsymbol{x}$ admits a subtour on some $t \geq 2$ cities that do not involve city 1. Let this subtour be $p_1, p_2, ..., p_t, p_1$. Hence, because $x_{p_1p_2} = x_{p_2p_3} = ... = x_{p_{t-2},p_{t-1}} = 1$, we get by using each of these in (3.1h) that,

$$y_{p_1p_t} = y_{p_2p_t} = y_{p_3p_t} = ... = y_{p_{t-1},p_t} \ . \tag{3.3}$$

Since $x_{p_{t-1},p_t} = 1$ implies from (3.1d) that $y_{p_{t-1},p_t} = 1$, we get from (3.3) that $y_{p_1p_t} = 1$. But $x_{p_tp_1} = 1$ implies from (3.1d) that $y_{p_tp_1} = 1$ as well, thus contradicting (3.1e). This completes the proof. □

3.2 RLT Lifting of Formulation ATSP0

We use a selective application of the first-order reformulation-linearization technique (RLT) process of Sherali and Adams [63,64] to tighten the formulation ATSP0. The RLT procedure is designed to generate a hierarchy of successively tighter linear programming (LP) approximations leading from the ordinary continuous relaxation to the convex hull representation for the mixed-binary programming problems.

The RLT procedure involves two basic steps: reformulation and linearization. In the reformulation step, it generates redundant, nonlinear inequalities by multiplying the constraints with the binary variables and their complements in a mixed, 0-1 linear program. Then, in the linearization step, it recasts the problem into a higher solution space by replacing each distinct product with a continuous variable. Here, we consider the first-order RLT process to tighten the formulation ATSP0 by generating a Level-1 formulation in concert with applying additional logical relationships among the resulting RLT product-variables as noted below.

To begin with, consider the following special RLT product constraints using (3.1b), (3.1c), and the implied constraint $y_{vj} \le 1,\ \forall v, j = 2,...,n,\ v \ne j$ (from (3.1e,i)):

$$\text{(i)}\quad \left(0 \le y_{vj} \le 1\right) * x_{iv},\ \forall i, j, v = 2,...,n,\ i \ne j \ne v. \tag{3.4a}$$

$$\text{(ii)}\quad \left[\sum_{v=1, v\ne i}^{n} x_{iv} = 1\right] * y_{ij},\ \forall i, j = 2,...,n,\ i \ne j. \tag{3.4b}$$

$$\text{(iii)}\quad \left[\sum_{i=1, i\ne v}^{n} x_{iv} = 1\right] * y_{vj},\ \forall v, j = 2,...,n,\ v \ne j. \tag{3.4c}$$

To linearize the product terms thus created, we primarily use the substitution

$$f_{ij}^{v} = x_{iv} \cdot y_{vj},\ \forall i, j, v = 2,...,n,\ i \ne j \ne v. \tag{3.5}$$

Next, we recognize the following identities as implied by the constraints defining ATSP0.

Proposition 3.2. The constraints defining ATSP0 imply the following identities, where the f-variables are as defined by (3.5).

$$x_{1v} \cdot y_{vj} = x_{1v}, \ \forall v, j = 2,...,n, \ v \neq j. \tag{3.6a}$$

$$x_{iv} \cdot y_{ij} = f_{ij}^{v}, \ \forall i, j, v = 2,...,n, \ i \neq j \neq v. \tag{3.6b}$$

$$x_{iv} \cdot y_{iv} = x_{iv}, \ \forall i, v = 2,...,n, \ i \neq v. \tag{3.6c}$$

$$x_{iv} \cdot y_{vi} = 0, \forall i, v = 2,...,n, \ i \neq v. \tag{3.6d}$$

$$x_{i1} \cdot y_{ij} = 0, \ \forall i, j = 2,...,n, \ i \neq j. \tag{3.6e}$$

Proof. If $x_{1v} = 0$, then (3.6a) is trivially true, and if $x_{1v} = 1$, then we must have $y_{vj} = 1$ by (3.1f). This establishes (3.6a). Next, observe that for any $i, j, v = 2,...,n$, $i \neq j \neq v$, we have

$$x_{iv} \cdot \left(y_{ij} - y_{vj}\right) = 0 \tag{3.7}$$

because this is trivially true when $x_{iv} = 0$, and when $x_{iv} = 1$, this is again true by virtue of (3.1h). Hence, using (3.5) in (3.7), we get (3.6b). Constraint (3.6c) is trivially true when $x_{iv} = 0$, and is also true when $x_{iv} = 1$, whence from (3.1d), we have $y_{iv} = 1$ as well. Note that (3.6c) and (3.1e) imply that $x_{iv} \cdot \left(1 - y_{vi}\right) = x_{iv}$, or that (3.6d) holds true. Finally, (3.6e) trivially holds true when $x_{i1} = 0$ and also when $x_{i1} = 1$, whence from (3.1g) and (3.1e) we get $y_{ji} = 1$ and $y_{ij} = 0$. This completes the proof. □

Applying (3.5) and (3.6a-e) of Proposition 3.2 to the RLT product constraints (3.4a,b,c), we get the following respective linearized constraints.

$$0 \leq f_{ij}^{v} \leq x_{iv}, \ \forall i, j, v = 2,...,n, \ i \neq j \neq v. \tag{3.8a}$$

$$\sum_{v=2, v \notin \{i,j\}}^{n} f_{ij}^{v} + x_{ij} = y_{ij}, \ \forall i, j = 2,...,n, \ i \neq j. \tag{3.8b}$$

$$x_{1v} + \sum_{i=2, i \notin \{v,j\}}^{n} f_{ij}^{v} = y_{vj}, \ \forall v, j = 2,...,n, \ v \neq j. \tag{3.8c}$$

Note that (3.8a) is obtained by directly substituting (3.5) into (3.4a), (3.8b) is obtained by using (3.6b,c,e) in (3.4b), and (3.8c) is obtained by applying (3.6a,d) along with (3.5) in (3.4c).

Augmenting ATSP0 with (3.8a,b,c) leads to the following tightened RLT-lifted reformulation, where we have omitted (3.1d), noting that this is now implied by (3.8b).

ATSP1: Minimize $\left\{ \sum_{i=1}^{n} \sum_{j=1, j\neq i}^{n} c_{ij} x_{ij} : \text{(3.1b,c), (3.1e)-(3.1i), and (3.8a,b,c)} \right\}$.

In our analysis below, we shall also consider the following relaxation of ATSP1 obtained by deleting (3.1f,g,h):

ATSP2: Minimize $\left\{ \sum_{i=1}^{n} \sum_{j=1, j\neq i}^{n} c_{ij} x_{ij} : \text{(3.1b,c,e,i), and (3.8a,b,c)} \right\}$.

We also consider another relaxation of ATSP1 by replacing (3.1h) with (3.2):

ATSP3: Minimize $\left\{ \sum_{i=1}^{n} \sum_{j=1, j\neq i}^{n} c_{ij} x_{ij} : \text{(3.1b,c,e,f,g,i), (3.2), and (3.8a,b,c)} \right\}$.

Note that ATSP3 is a valid model for ATSP by Sarin *et al.* [61] as discussed above, and the validity of ATSP2 is established in the sequel (see Corollary 3.1 in Section 3.3).

Note that the set of three-city subtour elimination constraints (3.2) in ATSP-SSB is respectively replaced by (3.9a) and (3.9b) given below to derive the corresponding models ATSP-SSB1 and ATSP-SSB2:

$$\left(y_{ij} + x_{ji}\right) + y_{jk} + y_{ki} \leq 2, \ \forall i, j, k = 2,...,n, \ i \neq j \neq k. \tag{3.9a}$$

$$x_{ij} + \left(y_{jk} + x_{kj}\right) + \left(y_{ki} + x_{ik}\right) \leq 2, \ \forall i, j, k = 2,...,n, \ i \neq j \neq k. \tag{3.9b}$$

Also note that, ATSP-SSB1 and ATSP-SSB2 contain an additional set of constraints (3.9c) below (see [61]) to further tighten the LP relaxation:

$$x_{1j} + x_{j1} \leq 1, \ \forall j = 2,...,n. \tag{3.9c}$$

We consider several augmentations to derive various tightened reformulations of ATSP-SSB1 and ATSP-SSB2. The first variation is to add constraint sets (3.1f) and (3.1g) to ATSP-SSB1 to obtain:

ATSP4: Minimize $\left\{ \sum_{i=1}^{n} \sum_{j=1, j\neq i}^{n} c_{ij} x_{ij} : \text{(3.1b)-(3.1g), (3.1i), and (3.9a,c)} \right\}$.

The second variation to tighten ATSP-SSB1 is to add to it the constraints (3.8a)-(3.8c). We designate the resulting formulation by:

ATSP5: Minimize $\left\{ \sum_{i=1}^{n} \sum_{j=1, j\neq i}^{n} c_{ij} x_{ij} : \text{(3.1b,c,d,e,i), (3.8a,b,c), and (3.9a,c)} \right\}$.

Moreover, we also tighten ATSP-SSB1 by adding the sets of constraints (3.1f,g) and (3.8a-c). As a result, (3.9c) is now omitted since it is implied by (3.1f,g), and (3.1d) is omitted since it is implied by (3.8b). The resulting formulation, designated ATSP6, turns out to be the same as ATSP3 except that (3.2) is replaced by (3.9a).

ATSP6: Minimize $\left\{ \sum_{i=1}^{n} \sum_{j=1, j\neq i}^{n} c_{ij} x_{ij} : \text{(3.1b,c,e,f,g,i), (3.8a,b,c), and (3.9a)} \right\}$.

The counterparts of the above variations for ATSP-SSB2 are as follows:

ATSP7: Minimize $\left\{ \sum_{i=1}^{n} \sum_{j=1, j\neq i}^{n} c_{ij} x_{ij} : \text{(3.1b)-(3.1g), (3.1i), and (3.9b,c)} \right\}$,

ATSP8: Minimize $\left\{ \sum_{i=1}^{n} \sum_{j=1, j\neq i}^{n} c_{ij} x_{ij} : \text{(3.1b,c,d,e,i), (3.8a,b,c), and (3.9b,c)} \right\}$,

and

ATSP9: Minimize $\left\{ \sum_{i=1}^{n} \sum_{j=1, j\neq i}^{n} c_{ij} x_{ij} : \text{(3.1b,c,e,f,g,i), (3.8a,b,c), and (3.9b)} \right\}$.

In the following section, we present results regarding the tightness of the polyhedrons related to the ***x***-variables of these formulations and also compare these with those of the mathematical formulations presented in the literature.

3.3 Dominance Relationship Among Various ATSP Formulations

First consider a tightening of ATSP-FL (see (2.2) in Section 2.1) in which we adopt the following additional valid restrictions, recognizing that p_{ij}^u represents the flow on arc (i, j) for the single unit of commodity u, which flows from node 1 to node u.

$$p_{j1}^u = 0\,,\ \forall j, u = 2,...,n\,. \tag{3.10a}$$

$$p_{uj}^u = 0\,,\ \forall j, u = 2,...,n\,,\ j \neq u\,. \tag{3.10b}$$

$$p_{1j}^u = x_{1j}\,,\ \forall j, u = 2,...,n\,. \tag{3.10c}$$

$$p_{ju}^u = x_{ju}\,, \forall j, u = 2,...,n\,,\ j \neq u\,. \tag{3.10d}$$

Observe that by making the substitutions (3.10) into the original formulation of ATSP-FL, we can make the following simplifications. First, by virtue of (3.10a,c), we have that (2.2e) reproduces (2.2c) for $i = 1$, and hence, (2.2e) can be deleted. Similarly, using (3.10a-d) in (2.2g) reproduces (2.2b) for $j = u$; hence, (2.2g) can also be omitted. Next, observe that (2.2d) for $i = 1$, or $j = 1$, or $i = u$, or $j = u$ are, respectively, implied by (3.10c), (3.10a), (3.10b), and (3.10d). Hence, (2.2d) needs to be written only for $i, j, u = 2,...,n\,,\ i \neq j \neq u$. Finally, applying (3.10a-d), we can rewrite (2.2f) in the following strengthened form:

$$\left[\sum_{j=2,\, j\notin\{i,u\}}^{n} p_{ij}^u + x_{iu}\right] - \left[\sum_{j=2,\, j\notin\{i,u\}}^{n} p_{ji}^u + x_{1i}\right] = 0\,,\ \forall i, u = 2,...,n\,,\ i \neq u\,. \tag{3.11}$$

Using these relationships in (2.2a-h), we can reformulate ATSP-FL as the following tightened version.

ATSP-FL2: Minimize $\sum_{i=1}^{n}\sum_{j=1,\, j\neq i}^{n} c_{ij}x_{ij}$ (3.12a)

subject to

$$\sum_{i=1, i\neq j}^{n} x_{ij} = 1\ ,\ \forall j = 1,...,n \tag{3.12b}$$

$$\sum_{j=1,\, j\neq i}^{n} x_{ij} = 1 \ , \ \forall i = 1,...,n \tag{3.12c}$$

$$p_{ij}^{u} \leq x_{ij} \ , \ \forall i, j, u = 2,...,n \ , \ i \neq j \neq u \tag{3.12d}$$

$$\left[\sum_{j=2,\, j\notin\{i,u\}}^{n} p_{ij}^{u} + x_{iu} \right] - \left[\sum_{j=2,\, j\notin\{i,u\}}^{n} p_{ji}^{u} + x_{1i} \right] = 0 \ , \ \forall i, u = 2,...,n \ , \ i \neq u \tag{3.12e}$$

$$x_{ij} \in \{0,1\} \ , \ \forall i, j = 1,...,n \ , \ i \neq j \tag{3.12f}$$

$$p_{ij}^{u} \geq 0 \ , \ \forall i, j, u = 2,...,n \ , \ i \neq j \neq u \ . \tag{3.12g}$$

Now, consider the following dominance result.

Proposition 3.3. ATSP2 affords a tighter LP relaxation than ATSP-FL2.

Proof. Note that the constraints (3.12b,c,f) of ATSP-FL2 are also present in ATSP2. Furthermore, let us rename the $\boldsymbol{p}$-variables in ATSP-FL2 as follows

$$p_{ij}^{u} = f_{iu}^{j} \ , \ \forall i, j, u = 2,...,n \ , \ i \neq j \neq u \ . \tag{3.13}$$

Then, (3.12d) and (3.12e) can be respectively re-written as follows:

$$f_{iu}^{j} \leq x_{ij} \ , \ \forall i, j, u = 2,...,n \ , \ i \neq j \neq u \ . \tag{3.14a}$$

$$\left[\sum_{j=2,\, j\notin\{i,u\}}^{n} f_{iu}^{j} + x_{iu} \right] - \left[\sum_{j=2,\, j\notin\{i,u\}}^{n} f_{ju}^{i} + x_{1i} \right] = 0 \ , \ \forall i, u = 2,...,n \ , \ i \neq u \ . \tag{3.14b}$$

Now, observe that (3.14a) is precisely (3.8a), and that (3.14b) is obtained by considering (3.8b) written for y_{iu} on its right-hand-side and subtracting from this (3.8c) written for the same y_{iu} on its corresponding right-hand-side. Hence, ATSP-FL2 is a relaxed surrogate of ATSP2, and this completes the proof. □

Corollary 3.1. The model ATSP2 affords a valid formulation of the asymmetric traveling salesman problem (ATSP).

Proof. The constraints defining ATSP2 are valid for ATSP since ATSP2 is a relaxation of ATSP1, which is a valid formulation by Proposition 3.1 and the RLT lifting process. Moreover, by Proposition 3.3, the constraints defining ATSP2 imply those of ATSP-FL2, which is a known valid formulation of ATSP. This completes the proof. □

For any model F for ATSP, let

$$\text{LP(F)} = \{\boldsymbol{x} : \ \boldsymbol{x} \text{ is part of a feasible solution to the LP relaxation of F}\}. \quad (3.15)$$

Then, we have the following string of dominance relationships, where ATSP-DFJ is the Dantzig-Fulkerson-Johnson formulation of ATSP (which has an exponential number of subtour elimination constraints), ATSP-MTZ is the Miller-Tucker-Zemlin formulation of ATSP, and where $conv(\text{ATSP})$ denotes the convex hull of binary vectors $\boldsymbol{x}$ that represent Hamiltonian circuits for ATSP.

Proposition 3.4. $conv(\text{ATSP}) \subseteq \text{LP}(\text{ATSP1}) \subseteq \text{LP}(\text{ATSP2}) \subseteq \text{LP}(\text{ATSP-FL2})$

$$\subseteq \text{LP}(\text{ATSP-FL}) = \text{LP}(\text{ATSP-DFJ}) \subseteq \text{LP}(\text{ATSP-MTZ}). \quad (3.16a)$$

Proof. Note that Wong [75] has established that $\text{LP}(\text{ATSP-FL}) = \text{LP}(\text{ATSP-DFJ})$, and it is well-known that $\text{LP}(\text{ATSP-DFJ}) \subseteq \text{LP}(\text{ATSP-MTZ})$ (see Nemhauser and Wolsey [54], for example). The other dominance relationships follow from the derivation of ATSP-FL2 from ATSP-FL, Proposition 3.3, the relaxation of ATSP1 to obtain ATSP2, the derivation of ATSP1 from ATSP0, and Proposition 3.1. □

Remark 3.1. Observe from the proof of Proposition 3.3 that the $\boldsymbol{f}$ -variables that appear in models ATSP1 and ATSP2 have a flow interpretation as prompted by (3.5) and (3.13). Namely, f_{ij}^{v}, which represents the nonlinear product $x_{iv} \cdot y_{vj}$, has the connotation of the flow on arc (i, v) of the single unit of commodity that effectively flows from the source node 1 to node j.

Remark 3.2. Desrochers and Laporte [20] have derived a tightened version (ATSP-DL) of ATSP-MTZ, and Sherali and Driscoll [67] have derived a further strengthened lifting (ATSP-SD) of this formulation, which has been exhibited to yield improved results on more challenging instances of ATSP and its precedence constrained version (PC-ATSP). The relationships among these formulations are as follows

$$\text{LP}(\text{ATSP-SD}) \subseteq \text{LP}(\text{ATSP-DL}) \subseteq \text{LP}(\text{ATSP-MTZ}). \quad (3.16b)$$

Furthermore, Sarin *et al.* [61] and Gouveia and Pires [26,27] have proposed a variety of polynomial length reformulations of the ATSP. Let us denote the best of these as ATSP-SSB2 (which is a lifting of ATSP-SSB referred to earlier), and ATSP-GP, respectively. Recalling that ATSP3 is an RLT-based lifting of ATSP-SSB2, we have the following known relationships from above:

$$\text{LP}(\text{ATSP3}) \subseteq \text{LP}(\text{ATSP-SSB2}) \subseteq \text{LP}(\text{ATSP-GP}) \subseteq \text{LP}(\text{ATSP-MTZ}). \qquad (3.16c)$$

Now, consider a relaxation ATSP2R of ATSP2 in which the constraints (3.1e) are replaced by

$$y_{ij} + y_{ji} \leq 1, \ \forall i, j = 2,...,n, \ i \neq j. \qquad (3.17)$$

Furthermore, denote by ATSP2R$^-$ the model obtained by deleting (3.1e) from ATSP2. We have the following result.

Proposition 3.5. Both ATSP2R and ATSP2R$^-$ are valid formulations of ATSP for which $\text{LP}(\text{ATSP2}) \subseteq \text{LP}(\text{ATSP2R}) \subseteq \text{LP}(\text{ATSP2R}^-) = \text{LP}(\text{ATSP-FL2})$. (3.18)

Proof. Note that in the model ATSP2R$^-$, the only constraints involving the $\boldsymbol{y}$-variables are the identities (3.8b) and (3.8c). These variables can be eliminated from ATSP2R$^-$ by simply equating the corresponding identities (3.8b) and (3.8c) for each of the $\boldsymbol{y}$-variables. But as shown in the proof of Proposition 3.3, this produces (3.14b), and hence, ATSP2R$^-$ is the same as ATSP-FL2. The remaining containment in (3.18) is obvious by construction. Furthermore, following the proof of Corollary 3.1 and using (3.18), ATSP2R and ATSP2R$^-$ are valid formulations of ATSP. This completes the proof. □

3.4 Computational Comparison of LP Relaxations of the Various ATSP Formulations

In this section, we present computational results to compare the LP relaxation bounds of our new formulations ATSP0, ATSP1, ATSP2, ATSP2R, ATSP2R$^-$ (which coincides with the tightened version ATSP-FL2 of ATSP-FL by the proof of Proposition 3.5), ATSP3, ATSP4, ATSP5, ATSP6, ATSP7, ATSP8, and ATSP9 with those of nine other models, ATSP-SSB1, ATSP-SSB2, ATSP-SD, ATSP-FL, ATSP-FL2, ATSP-GP, ATSP-GP1, ATSP-GP2 and ATSP-GP3. Note that ATSP-SSB2 has been shown in [25] to provide tighter lower bounds than ATSP-SSB and ATSP-SSB1. Moreover, ATSP-GP (see Gouveia and Pires [26,27]) includes constraint sets (2b,c,d) with the additional logical relations (3.19a,b) as follows:

$$x_{ij} + y_{ji} \le 1,\ \forall i, j = 2,...,n,\ i \ne j. \tag{3.19a}$$

$$x_{ij} + y_{ki} \le y_{kj} + 1,\ \forall i, j, k = 2,...,n,\ i \ne j \ne k. \tag{3.19b}$$

Two lifted versions of ATSP-GP, namely, ATSP-GP1 and ATSP-GP2, replace (3.19b) by (3.19c) and (3.19d), respectively, as given below:

$$x_{ij} + x_{ji} + y_{ki} \le y_{kj} + 1,\ \forall i, j, k = 2,...,n,\ i \ne j \ne k. \tag{3.19c}$$

$$x_{ij} + x_{kj} + x_{ik} + y_{ki} \le y_{kj} + 1,\ \forall i, j, k = 2,...,n,\ i \ne j \ne k. \tag{3.19d}$$

ATSP-GP3 is another lifted version of ATSP-GP obtained by replacing (3.19b) with both (3.19c) and (3.19d). Also, similar to ATSP-SSB1 and ATSP-SSB2, we included the set of two-city subtour constraints (3d) in each of ATSP-GP1, ATSP-GP2, and ATSP-GP3 in order to further tighten their LP relaxations.

Six standard test problems are selected from the TSP library (TSPLIB), namely, br17, ftv33, ftv35, ftv38, p43, and ftv44. The number appended to each of these test cases indicates the number of cities that it contains; hence, br17 is an asymmetric traveling salesman problem having 17 cities. All runs were made on a Dell Workstation PWS 650

with double 2.5GHz CPU Xeon processors and 1.5GB RAM, running the Windows XP operating system. We used AMPL version 8.1 and CPLEX MIP Solver version 9.0.

The results shown in Table 3-1 include the lower bound value (z_{LP}) obtained by solving the LP relaxation and the cpu time (in seconds) required to obtain the optimal LP solution. Note that the lower bounds (z_{LP}) obtained via the LP relaxations of ATSP1, ATSP2, ATSP2R, ATSP2R$^-$, ATSP3, ATSP5, ATSP6, ATSP8 and ATSP9 are at least as tight as that obtained by solving the LP relaxations of ATSP-FL and ATSP-FL2. Furthermore, ATSP5 and ATSP6 produced the tightest lower bounds, and ATSP3 yielded the second tightest lower bound values for all the test problems.

Recall that ATSP3, ATSP5 and ATSP6 enforce the three-city subtour elimination constraints (3.2) on the $\boldsymbol{y}$-variables; moreover, ATSP5 and ATSP6 replace (3.2) within ATSP3 by the lifted version (3.9a). The third tightest lower bound values are obtained using ATSP1, ATSP2, ATSP8 and ATSP9. The commonalities of constraint sets in these formulations are (3.1e) and (3.8a)-(3.8c).

While comparing the results of ATSP8 and ATSP-SSB2, the lower bound value obtained due to the addition of constraint sets (3.8a)-(3.8c) is either the same or tighter but at the cost of increased computation time. Also, note that, the computation time required for solving the LP relaxation of ATSP-FL2 is about the same or less than that required for solving ATSP-FL, while both of these formulations produce identical LP bound values for all of our test cases. A comparison of the results of ATSP7 and ATSP-SSB2 reveals that the addition of constraint sets (3.1f,g) results in a possible increment in the computation time for solving the LP relaxation of ATSP7 (over ATSP-SSB2) while giving the same lower bound values for all test cases.

Table 3-1 Computational results pertaining to LP relaxations of the various ATSP formulations

Problem	Formulation	z_{LP}	CPU time (LP)	Problem	Formulation	z_{LP}	CPU time (LP)
br17 (z_{IP}=39)	ATSP-SSB2	28.00	0.16	**ftv38** (z_{IP}=1503)	ATSP-SSB2	1484.00	44.85
	ATSP-SD	27.68	0.05		ATSP-SD	1458.22	0.78
	ATSP-FL	**39.00**	1.63		ATSP-FL	1482.00	100.12
	ATSP-FL2	**39.00**	0.86		ATSP-FL2	1482.00	52.16
	ATSP-GP	18.00	0.03		ATSP-GP	1459.50	3.63
	ATSP-GP1	22.00	0.02		ATSP-GP1	1461.08	6.66
	ATSP-GP2	28.00	0.06		ATSP-GP2	1481.29	6.81
	ATSP-GP3	28.00	0.15		ATSP-GP3	1483.00	28.26
	ATSP0	22.00	0.22		ATSP0	1459.50	41.89
	ATSP1	**39.00**	19.56		ATSP1	1486.10	47,445.70
	ATSP2	**39.00**	5.81		ATSP2	1486.10	14,426.60
	ATSP2R	**39.00**	0.50		ATSP2R	1482.00	239.13
	ATSP2R$^-$	**39.00**	1.67		ATSP2R$^-$	1482.00	110.61
	ATSP3	**39.00**	10.55		ATSP3	1486.10	39,336.40
	ATSP4	22.00	0.36		ATSP4	1463.50	92.42
	ATSP5	**39.00**	12.98		ATSP5	**1488.22**	23,127.40
	ATSP6	**39.00**	10.88		ATSP6	**1488.22**	8,953.23
	ATSP7	28.00	0.17		ATSP7	1484.00	57.89
	ATSP8	**39.00**	14.91		ATSP8	1486.10	10,630.70
	ATSP9	**39.00**	12.66		ATSP9	1486.10	30,898.60
ftv33 (z_{IP}=1286)	ATSP-SSB2	**1286.00**	17.03	**p43** (z_{IP}=5620)	ATSP-SSB2	241.22	123.58
	ATSP-SD	1224.50	0.42		ATSP-SD	864.58	1.49
	ATSP-FL	**1286.00**	49.53		ATSP-FL	**5611.00**	2,392.55
	ATSP-FL2	**1286.00**	26.23		ATSP-FL2	**5611.00**	915.88
	ATSP-GP	1224.68	1.81		ATSP-GP	184.00	3.17
	ATSP-GP1	1226.25	3.08		ATSP-GP1	216.00	9.75
	ATSP-GP2	**1286.00**	3.83		ATSP-GP2	241.14	7.06
	ATSP-GP3	**1286.00**	13.45		ATSP-GP3	242.00	45.50
	ATSP0	1224.68	23.67		ATSP0	216.00	148.96
	ATSP1	**1286.00**	16,536.60		ATSP1	**5611.00**	357,313.00
	ATSP2	**1286.00**	248.87		ATSP2	**5611.00**	37,141.00
	ATSP2R	**1286.00**	34.70		ATSP2R	**5611.00**	12,346.60
	ATSP2R$^-$	**1286.00**	27.55		ATSP2R$^-$	**5611.00**	3,094.75
	ATSP3	**1286.00**	7,147.47		ATSP3	**5611.00**	32,836.30
	ATSP4	1229.08	43.78		ATSP4	216.00	232.87
	ATSP5	**1286.00**	8,095.53		ATSP5	**5611.00**	57,308.40
	ATSP6	**1286.00**	5,540.85		ATSP6	**5611.00**	41,386.80
	ATSP7	**1286.00**	9.09		ATSP7	241.22	120.09
	ATSP8	**1286.00**	26,271.20		ATSP8	**5611.00**	179,775.00
	ATSP9	**1286.00**	6,912.45		ATSP9	**5611.00**	50,721.10
ftv35 (z_{IP}=1473)	ATSP-SSB2	1456.89	26.77	**ftv44** (z_{IP}=1613)	ATSP-SSB2	1584.88	136.65
	ATSP-SD	1415.51	0.34		ATSP-SD	1573.75	0.14
	ATSP-FL	1457.33	81.01		ATSP-FL	1584.88	262.13
	ATSP-FL2	1457.33	46.52		ATSP-FL2	1584.88	211.42
	ATSP-GP	1422.86	2.25		ATSP-GP	1578.83	10.31
	ATSP-GP1	1426.90	4.91		ATSP-GP1	1582.00	17.73
	ATSP-GP2	1456.06	5.55		ATSP-GP2	1584.88	14.50
	ATSP-GP3	1456.69	14.14		ATSP-GP3	1584.88	68.64
	ATSP0	1424.57	23.95		ATSP0	1580.17	316.18
	ATSP1	1460.20	48,198.30		ATSP1	1590.74	158,416.00
	ATSP2	1460.20	1,448.53		ATSP2	1590.74	42,928.20
	ATSP2R	1457.33	61.02		ATSP2R	1584.88	1,528.93
	ATSP2R$^-$	1457.33	86.12		ATSP2R$^-$	1584.88	277.81
	ATSP3	1460.20	10,876.10		ATSP3	1590.91	109,782.00
	ATSP4	1427.00	44.14		ATSP4	1582.00	417.41
	ATSP5	**1463.41**	10,588.00		ATSP5	**1594.94**	69,517.90
	ATSP6	**1463.41**	19,271.70		ATSP6	**1594.94**	159,105.00
	ATSP7	1456.89	8.91		ATSP7	1584.88	150.80
	ATSP8	1460.20	13,839.90		ATSP8	1590.74	81,880.40
	ATSP9	1460.20	15,583.00		ATSP9	1590.74	169,472.00

3.5 Results for Solving the Lagrangian Dual Formulations of the LP Relaxations by Using Nondifferentiable Optimization Algorithms

As shown in Section 3.4, ATSP5 and ATSP6 provide the tightest LP-based lower bounds for the ATSP problems. However, the cpu times required to solve the LP-relaxations of these formulations are high for large-size problems. The ineffectiveness of the simplex method could be the ill-conditioned primal problem resulting from the LP relaxation. Hence, in this section, we consider solution of the LP relaxations of these ATSP formulations via Lagrangian dual (LD) reformulation.

To introduce the setup of a Lagrangian dual formulation, consider the following mixed integer programming problem.

MIP: Minimize $c^T x$

Subject to $Ax \geq b$

$$x \in X_I = \left\{ \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \in \Re^n : Dx \geq d, \quad 0 \leq x_1 \leq u, \quad x_2 \in \{0,1\}^{n_2} \right\}$$

where $x_1 \in \Re^{n_1}$, $x_2 \in \{0,1\}^{n_2}$, $n = n_1 + n_2$, A is $n \times m$, and D is $l \times n$. The dimensions of b, c, d, and u are in accordance with those of x_1, x_2, A and D, respectively.

The linear programming relaxation of MIP is given by

LR: Minimize $c^T x$

Subject to $Ax \geq b$

$$x \in X = \left\{ \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \in \Re^n : Dx \geq d, \quad 0 \leq x_1 \leq u, \quad 0 \leq x_2 \leq e \right\}$$

Then, after dualizing the set of constraints $Ax \geq b$, we have the following Lagrangian Dual problem for LR.

LD: Maximize $\theta(\pi)$

Subject to $\pi \geq 0$

where $\theta(\pi) = \min\{c^T x + \pi^T (b - Ax) : x \in X\}$. Note that $\pi \in \Re^m$ is the dual variable vector associated with the set of constraint $Ax \geq b$.

To investigate the performance of this method, we consider the LP relaxation of ATSP6 as follows.

Maximize $\theta_{\text{ATSP6}}(\pi)$

Subject to $\pi^1, \pi^2, \pi^3, \pi^7, \pi^8$ unrestricted; $\pi^4, \pi^5, \pi^6, \pi^9 \geq 0$,

where $\theta_{\text{ATSP6}}(\pi) = \underset{x,y,f\geq 0}{\text{Minimize}} \left\{ \left(\sum_{i=1}^{n} \sum_{j=1, j\neq i}^{n} c_{ij} x_{ij} + \sum_{i=1}^{n} \pi_i^1 \cdot \left(1 - \sum_{v=1, v\neq i}^{n} x_{iv} \right) \right.\right.$

$+ \sum_{v=1}^{n} \pi_v^2 \cdot \left(1 - \sum_{i=1, i\neq v}^{n} x_{iv} \right) + \sum_{i,j=2, i\neq j}^{n} \pi_{ij}^3 \cdot (1 - y_{ij} - y_{ji}) + \sum_{i,j=2, i\neq j}^{n} \pi_{ij}^4 \cdot (x_{1i} - y_{ij})$

$+ \sum_{i,j=2, i\neq j}^{n} \pi_{ij}^5 \cdot (x_{i1} - y_{ji}) - \sum_{i,j,v=2, i\neq j\neq v}^{n} \pi_{ijv}^6 \cdot (x_{iv} - f_{ij}^v) + \sum_{i,j=2, i\neq j}^{n} \pi_{ij}^7 \cdot \left(y_{ij} - x_{ij} - \sum_{v=2, v\notin\{i,j\}}^{n} f_{ij}^v \right)$

$\left. + \sum_{v,j=2, v\neq j}^{n} \pi_{jv}^8 \cdot \left(y_{vj} - x_{1v} - \sum_{i=2, i\notin\{v,j\}}^{n} f_{ij}^v \right) - \sum_{i,j,v=2, i\neq j\neq v}^{n} \pi_{ijv}^9 \cdot (2 - (y_{ij} + x_{ji}) - y_{jv} - y_{vi}) \right) \right\}$

Assuming that either an optimal objective value or a good upper bounding solution value from some heuristic algorithm is given, we solve this LD reformulation by a subgradient algorithm. In lieu of using pure subgradient method, we apply average direction strategy (ADS) as a subgradient deflection method to improve its convergence and efficiency [69]. The subgradient algorithm with average direction strategy (SA_ADS) is presented next. But, first, we introduce the relevant notation.

π^*, θ^* ≡ incumbent solution and objective value.

ξ^* ≡ incumbent subgradient vector.

$\theta(UB)$ ≡ given optimal or heuristic upper bounding objective value.

ε_0, ε ≡ termination tolerance on subgradient norm and the tolerance for an acceptable (near-) optimal objective value (e.g. $\varepsilon_0 = 10^{-6}$, $\varepsilon = 10^{-4}$).

β_k ≡ step length parameter at iteration k, $\beta_k \in [\beta_{\min}, \beta_{\max}]$.

λ_k ≡ step size at iteration k, $\lambda_k > 0$.

N ≡ maximum number of iterations.

$\bar{r}$ ≡ number of iterations in each block.

$\bar{\nu}$ ≡ maximum number of consecutive iterations over which the current solution value fails to improve.

γ ≡ iteration counter of consecutive iterations over which the current solution value fails to improve.

h ≡ integer that indicates the number of times the step length parameter β_k should be halved.

Subgradient Algorithm with Average Direction Strategy (SA_ADS) [69]

Initialization Step:

Select $\varepsilon_0 \geq 0$, $\varepsilon > 0$, N, $\bar{r}$, $\bar{\nu}$, $\beta_{\min}$, and $\beta_{\max}$. Choose an initial solution π_1 and determine $\theta(\pi_1)$, ξ_1. Set $T = \left| N / \bar{r} \right|$, the incumbent solution $\pi^* = \pi_1$, $\theta^* = \theta(\pi_1)$, $\xi^* = \xi_1$, $k = 1$, $t = 1$, $h = 0$.

Main Step:

Given π_k, find a subgradient $\xi_k \in \partial\theta(\pi_k)$ of $\theta(\cdot)$ at π_k.

If $\|\xi_k\| \leq \varepsilon_0$ (a relaxed stopping criteria of $\xi_k = 0$) or $\theta^* \geq \theta(UB) - \varepsilon$, stop, π^* is optimal; otherwise, let the deflected direction $d_k = \xi_k + \dfrac{\|\xi_k\|}{\|d_{k-1}\|} \cdot d_{k-1}$.

Select a step size $\lambda_k = \dfrac{\beta_k \cdot [\theta(UB) - \theta(\pi_k)]}{\|d_k\|^2}$, where

$$\beta_k = \max\left(2^{-h} \cdot \left(\beta_{\max} + \frac{(t-1)}{(T-1)} * [\beta_{\min} - \beta_{\max}] \right), \varepsilon_0 \right).$$

Compute $\pi_{k+1} = \mathrm{P}_{\pi}(\bar{\pi}_{k+1})$, where $\bar{\pi}_{k+1} = \pi_k + \lambda_k \cdot d_k$.

If $\theta(\pi_{k+1}) > \theta^*$, set $\theta^* = \theta(\pi_{k+1})$ and $\pi^* = \pi_{k+1}$, $\xi^* = \xi_{k+1}$, $h = 0$, $\gamma = 0$; otherwise, increment $\gamma \leftarrow \gamma + 1$.

If $k = N$, stop and output π^*. If $\gamma \geq \bar{\nu}$, reset current solution to the incumbent solution and set $h \leftarrow h + 1$, $\gamma = 0$. If $k = t \cdot \bar{r}$, reset current solution to the incumbent solution and set $t \leftarrow t + 1$, $h = 0$, $\gamma = 0$. Let $k \leftarrow k + 1$ and repeat the main step.

The algorithm was coded and compiled on a Visual C++ compiler (version 6.0). All runs in this section were made on a Dell Workstation PWS 650 with double 2.5GHz CPU Xeon processors and 1.5GB RAM, running the Windows XP operating system. In this experiment, we set the maximum step length parameter $\beta_{\max} = 0.8$, the minimum step length parameter $\beta_{\min} = 10^{-6}$, and use an initial dual solution vector π_1 with each of its elements equal to 10^{-5}. Two test problems, namely br4 and br17, were solved to evaluate the above algorithm. Note that br4 is a reduced test problem that includes only the first 4 cities of br17. A primal problem can be expressed in a standard format along with the bounded variables (i.e, Minimize $\{ c^T x \ : \ Ax = b \, , l \le x \le u \}$) or in a canonical format along with the bounded variables (i.e, Minimize $\{ c^T x \ : \ Ax \ge b \, , l \le x \le u \}$). We generate both formats of the test problems using AMPL (version 8.1). Table 3-2 shows the sizes of these two problems in standard and canonical formats. This includes the number of variables, the number of constraints, the number of nonzero entities in matrix A, and the cpu time (in seconds) required to compute the optimal LP solution using the simplex method in CPLEX. The number of iterations in each block are set to $\bar{r} = 5$ and $\bar{r} = 10$, and maximum number of consecutive non-improving iterations are set to $\bar{\nu} = 2$ and $\bar{\nu} = 4$ for the test problem br4 and br17, respectively.

Table 3-2 Summary of test problems for br4 and br17

Problem	Format	Number of variables	Number of constraints	Number of nonzeros	CPU time (CPLEX)
br4 (z_{LP} =13)	Standard	48	50	156	0.01
	Canonical	24	50	132	0.01
br17 (z_{LP} =39)	Standard	11,072	7,954	37,024	14.92
	Canonical	3,872	7954	29,824	16.11

The results are shown in Table 3-3. It depicts the near optimal lower bound values (near-opt z_{LP}) derived by solving the Lagrangian dual of the LP relaxation of ATSP6, the cpu time (in seconds) required to compute this solution, and the number of iterations before it returns the optimal solution. The maximum number of iterations allowed (N) is also indicated for each problem. Note that in this computational experiment, the initial solution was improved by moving along the subgradient using a very small step size ($\lambda = 10^{-6}$) until no improvement was achieved within 60 iterations.

Table 3-3 Computational results for the solution of br4 and br17 using SA_ADS

Problem	N	Standard format			Canonical format		
		Near-opt z_{LP}	CPU time (s)	Iteration	Near-opt z_{LP}	CPU time (s)	Iteration
br4 (z_{LP}=13)	2,000	12.9999	0.031	152	12.9999	0.001	34
	5,000	12.9999	0.016	169	12.9999	0.016	34
	10,000	12.9422	2.046	10,000	12.9999	0.001	34
br17 (z_{LP}=39)	20,000	-0.02646	0.016	3	-0.01414	0.015	3
	50,000	-0.02646	0.016	3	-0.01414	0.015	3
	100,000	-0.02646	0.016	3	-0.01414	0.015	3

As is evident from Table 3-3, the subgradient method SA_ADS effectively solves the test problem br4 close to optimality. However, it fails to solve the LD reformulation for test problem br17 to (near-) optimality.

Since the upper bounding solution value might not be available, and also, the objective function of the Lagrangian dual formulation is nondifferentiable, we consider two nondifferentiable optimization (NDO) methods, namely, the variable target value method (VTVM) of Sherali *et al*. [66] and the trust region target value (TRTV) method of Lim and Sherali [46] for the solution of this problem. Both of these methods do not assume a priori knowledge of a bound on the optimal value, find (near-) optimal solutions iteratively by searching along an improving direction in each iteration, and use a perturbation technique when a nondifferentiable point is encountered. The difference between these two methods is in the way that the target value is updated. In the VTVM, the target value is increased (moved farther away from the incumbent solution) when a significant improvement is achieved, and is decreased (moved closed to the incumbent solution) when a specified effort has been spent without obtaining a significant improvement. The details of both of these procedures are given in Lim [43].

To implement the nondifferentiable optimization methods, we adapt the computer codes developed by Lim [44], which is compiled on a Visual C++ compiler (version 6.0) along with the CPLEX callable library (version 8.1). The following parameter values are considered in both of these algorithms: the number of iterations for the perturbation technique (K), the strategy for the perturbation technique (PT), the step-length parameter, and the number of iterations for the VTVM or for the TRTV method (k_max). Note that, the maximum number of cuts for the box trust region (BTR) is set to zero (i.e., cut_max =

0) and the step-length parameter is set to 0.8 (i.e., $\beta = 0.8$) in all test runs. Also, we consider two strategies for the perturbation technique, the Hestenes and Stiefel (HS) [32] strategy and the average direction strategy (ADS) [69].

Table 3-4 shows the near optimal lower bound values (near-opt z_{LP}) obtained by solving the Lagrangian dual of the LP relaxation of ATSP6 in standard format and the cpu time (in seconds) required to compute this solution for both the VTVM and the TRTV method.

Table 3-4 Computational results for solving br4 and br17 by the VTVM and TRTV method in standard format

Problem	K	PT	k_max	VTVM		TRTV	
				Near-opt z_{LP}	CPU time (s)	Near-opt z_{LP}	CPU time (s)
br4 (z_{LP}=13)	0	HS	200	12.2073	0.001	9.5067	0.032
			500	12.9840	0.031	11.7019	0.031
			1000	12.9998	0.078	12.8389	0.063
		ADS	200	12.2073	0.001	9.5067	0.031
			500	12.9840	0.031	11.7019	0.032
			1000	12.9998	0.078	12.8389	0.063
	20	HS	200	12.2073	0.015	9.5067	0.016
			500	12.9840	0.031	11.7019	0.047
			1000	12.9998	0.078	12.8389	0.078
		ADS	200	12.2073	0.016	9.5067	0.015
			500	12.9840	0.032	11.7019	0.046
			1000	12.9998	0.078	12.8389	0.062
br17 (z_{LP}=39)	0	HS	1000	4.6816	7.328	18.5932	4.672
			2000	4.7619	14.563	20.8205	9.359
			3000	4.7619	21.172	21.7784	13.937
		ADS	1000	4.6816	7.437	18.5932	4.688
			2000	4.7619	14.578	20.8205	9.359
			3000	4.7619	21.172	21.7784	13.968
	100	HS	1000	4.6816	11.047	18.5932	6.921
			2000	4.7619	19.829	20.8205	11.750
			3000	4.7619	28.078	21.7784	16.344
		ADS	1000	4.6816	11.032	18.5932	6.890
			2000	4.7619	19.844	20.8205	11.625
			3000	4.7619	28.000	21.7784	16.203

The variable target value method (VTVM) can solve a primal problem expressed in a standard format and including bounded variables or in a canonical format including bounded variables; while the trust region target value (TRTV) method can only consider a primal problem expressed in a standard format including bounded variables. Table 3-5 shows the near optimal lower bound values (near-opt z_{LP}) obtained by solving the

Lagrangian dual of the LP relaxation of ATSP6 in canonical form and the cpu time (in seconds) required to compute this solution using the VTVM.

Table 3-5 Computational results for solving br4 and br17 by the VTVM in canonical format

		br4 (z_{LP} =13)				br17 (z_{LP} =39)		
PT	K	k_max	Near-opt z_{LP}	CPU time (s)	K	k_max	Near-opt z_{LP}	CPU time (s)
HS	0	200	12.8949	0.001	0	1000	33.8216	6.141
		500	12.9986	0.031		2000	33.8701	13.219
		1000	13.0000	0.063		3000	33.8702	17.797
	20	200	12.8949	0.016	100	1000	33.8216	9.204
		500	12.9986	0.047		2000	33.8701	16.969
		1000	13.0000	0.078		3000	33.8702	24.094
ADS	0	200	12.8949	0.001	0	1000	33.8216	6.172
		500	12.9986	0.031		2000	33.8701	12.156
		1000	13.0000	0.062		3000	33.8702	17.797
	20	200	12.8949	0.015	100	1000	33.8216	9.219
		500	12.9986	0.047		2000	33.8701	16.766
		1000	13.0000	0.063		3000	33.8702	23.953

From the results depicted in Table 3-4 and Table 3-5, note that the near optimal values obtained by the VTVM, using the canonical format of the LP relaxation of ATSP6, is the tightest under the same parameter settings. When it is run for more than 2000 iterations for br17, the VTVM results in no improvement. Moreover, note that, the number of iterations for the perturbation technique (e.g. K=0, 20, or 100) does not seem to make any difference in the near optimal lower bound values (near-opt z_{LP}) obtained by solving the Lagrangian dual of the LP relaxation of ATSP6 using both NDO algorithms.

3.6 Implementation of the Strongest Surrogate Constraint Cuts in ATSP2R$^-$

In this section, we show how to derive a set of strong valid inequalities by using the tighter formulations, such as ATSP5 or ATSP6, through a suitable surrogation process and to use these cuts within a more compact and manageable formulation, such as ATSP2R$^-$. The idea is to include the relative strength in tightness of the formulations ATSP5 or ATSP6 in ATSP2R$^-$, a formulation that generally requires small cpu times to obtain the IP optimal (z_{IP}) solution.

We use the following steps to lift ATSP2R$^-$ using the strongest surrogate constraint generated from either ATSP5 or ATSP6.

Step 1: Solve the LP relaxation of ATSP6 (or ATSP5) and get the optimal LP solution.

Step 2: If ATSP6 is used, generate a constraint by surrogating constraint sets (3.1e)-(3.1g), (3.19a). Else, if ATSP5 is solved, generate a constraint by surrogating constraint sets (3.1d,e), (3.19a,c). Add this constraint to ATSP2R^{-}, denoting the resulting formulation as ATSP2R$^{\pm}$.

Step 3: Solve the LP relaxation of ATSP2R$^{\pm}$. Also, obtain its IP optimal solution, using the same setting as before.

We tested the above surrogation process on six standard test problems from the TSP library (TSPLIB), namely, br17, ftv33, ftv35, ftv38, p43, and ftv44. All runs were made on a Dell Workstation PWS 650 with double 2.5GHz CPU Xeon processors and 1.5GB RAM, running the Windows XP operating system. We used AMPL version 8.1 and CPLEX MIP Solver version 9.0.

The results are shown in Table 3-6. These include the lower bound value (z_{LP}) obtained by solving the LP relaxation, the cpu time (in seconds) required to obtain the optimal LP solution, the percentage gap between the LP-based lower bound and the optimal integer solution, z_{IP}, the cpu time (in seconds) required to obtain the best IP solution, and the number of branching nodes. While solving for the best integer solution, we limited the search to 10,000 cpu seconds. Note that the number in bold indicates the highest z_{LP} obtained among all models.

Note that the LP relaxation of ATSP2R$^{\pm}$ now obtains the best z_{LP} values with a slight increment in cpu time thus showing the effectiveness of the surrogate constraint. Regarding the IP optimal solution, the additional surrogate constraint does not seem to help in reducing the cpu time; in fact, it increases the number branching nodes generated but it improves the best z_{LP} values.

Table 3-6 Results on adding surrogate constraints from ATSP 5 or ATSP 6 to ATSP2R$^-$

Problem	Formulation	z_{LP}	CPU time (LP)	Gap (%)	Best Integer	CPU time (IP)	Nodes
br17 (z_{IP}=39)	ATSP2R$^-$	**39.00**	1.67	0.0	39	13.2	0
	ATSP5	**39.00**	11.95	0.0	39	55.7	0
	ATSP6	**39.00**	11.00	0.0	39	51.7	0
	ATSP2R$^{\pm}$ (ATSP5)	**39.00**	1.06	0.0	39	1.5	0
	ATSP2R$^{\pm}$ (ATSP6)	**39.00**	1.84	0.0	39	1.8	0
ftv33 (z_{IP}=1286)	ATSP2R$^-$	**1286.00**	27.55	0.0	1286	35.5	0
	ATSP5	**1286.00**	11,540.10	0.0	1286	0	0
	ATSP6	**1286.00**	5,513.55	0.0	1286	1,284.5	0
	ATSP2R$^{\pm}$ (ATSP5)	**1286.00**	38.67	0.0	1286	32.0	0
	ATSP2R$^{\pm}$ (ATSP6)	**1286.00**	38.45	0.0	1286	55.4	0
ftv35 (z_{IP}=1473)	ATSP2R$^-$	1457.33	86.12	1.1	1473	1,024.1	43
	ATSP5	**1463.41**	12,074.50	0.7	2036	10,000.0*	0
	ATSP6	**1463.41**	19,210.50	0.7	1806	10,000.0*	0
	ATSP2R$^{\pm}$ (ATSP5)	**1463.41**	274.74	0.7	1473	2,282.5	71
	ATSP2R$^{\pm}$ (ATSP6)	**1463.41**	272.95	0.7	1473	2,001.0	57
ftv38 (z_{IP}=1503)	ATSP2R$^-$	1482.00	110.61	1.4	1503	540.2	18
	ATSP5	**1488.22**	24,712.30	1.0	-**	10,000.0*	0
	ATSP6	**1488.22**	31,223.00	1.0	-**	10,000.0*	0
	ATSP2R$^{\pm}$ (ATSP5)	**1488.22**	333.90	1.0	1503	2,368.6	29
	ATSP2R$^{\pm}$ (ATSP6)	**1488.22**	378.19	1.0	1503	3,314.2	47
p43 (z_{IP}=5620)	ATSP2R$^-$	**5611.00**	1,138.02	0.2	5633	10,000.0*	0
	ATSP5	**5611.00**	36,272.80	0.2	-**	10,000.0*	0
	ATSP6	**5611.00**	34,998.10	0.2	-**	10,000.0*	0
	ATSP2R$^{\pm}$ (ATSP5)	**5611.00**	1,527.77	0.2	5664	10,000.0*	0
	ATSP2R$^{\pm}$ (ATSP6)	**5611.00**	1,658.79	0.2	5657	10,000.0*	0

- * Exceeded time limit.
- ** No integer solution was obtained within the time limit.

3.7 ATSP Formulations with Precedence Constraints

In this section, we extend our discussion to the precedence constrained asymmetric traveling salesman problem (PCATSP), also known as the sequential ordering problem (SOP). Let PC_j be the set of cities that are required to precede city j in PCATSP, $\forall j$. If $i \in PC_j$, then we denote this as $i \prec j$.

For the ATSP formulations that include the $(\boldsymbol{x}, \boldsymbol{y})$-variables, namely, ATSP0, ATSP1, ATSP2, ATSP2R, ATSP2R$^-$, ATSP3, ATSP4, ATSP5, ATSP6, ATSP7, ATSP8, and ATSP9, the precedence relationship can be captured explicitly by applying the definition of these variables as follows.

$$y_{ij} = 1,\ \forall i \in PC_j,\ j = 2,...,n. \tag{3.20a}$$

$$x_{kj} = 0\ ,\ \forall j = 2,...,n,\ \forall k \in PC_i \text{ where } i \in PC_j. \tag{3.20b}$$

Note that (3.20a), (3.1e), and (3.1d) imply that $y_{ji} = 0,\ \forall i \in PC_j,\ j = 2,...,n$ and that $x_{ji} = 0,\ \forall i \in PC_j,\ j = 2,...,n$. Whenever (3.1d,e) are not included in the models, we explicitly restrict:

$$x_{ji} = 0,\ \forall i \in PC_j,\ j = 2,...,n. \tag{3.20c}$$

Hence, for all the foregoing models, to formulate the corresponding PCATSP version, we include (3.20a,b), while for PCATSP2R and PCATSP2R$^-$, we add (3.20a)-(3.20c). From the analysis of Section 3.2, the resulting models are all valid formulations of PCATSP, and share the same dominance relationship as for the corresponding ATSP models.

Moreover, to accommodate the precedence relationship within ATSP-FL2, we can first equivalently write this problem as ATSP2R$^-$ following the proof of Proposition 3.5, and then add (3.20a)-(3.20c) as above. Alternatively, we can accomplish this directly within ATSP-FL2 without introducing the additional $\boldsymbol{y}$-variables as follows.

Observe from the interpretation of the multi-commodity flow variables p_{ij}^u in Section 3.3, that if $j \in PC_u$, then node j must lie on the flow-path from node 1 to node u in the flow of commodity u. This can be enforced by

$$\sum_{i=1, i\neq j}^{n} p_{ij}^u = 1,\ \forall j \in PC_u,\ \forall u = 2,...,n. \tag{3.21}$$

Using (13b,c), we can write (3.21) as

$$x_{1j} + \sum_{i=2, i\notin\{j,u\}}^{n} p_{ij}^u = 1,\ \forall j \in PC_u,\ \forall u = 2,...,n. \tag{3.22a}$$

In addition, we can restrict the $\boldsymbol{x}$-variables according to

$$x_{kj} = 0,\ \forall j = 2,...,n,\ \forall k \in PC_i, \text{ where } i \in PC_j. \tag{3.22b}$$

$$x_{ji} = 0,\ \forall i \in PC_j,\ \forall j = 2,...,n. \tag{3.22c}$$

Hence, to directly formulate PCATSP-FL2 without using the $\boldsymbol{y}$-variables, we accommodate (3.22a)-(3.22c) within ATSP-FL2.

Remark 3.3. Observe from (3.13) that we have the following interpretation relationships: $p_{ij}^{u} = f_{iu}^{j}$. Using this in (3.8c), we can equivalently write this constraint as $x_{1j} + \sum_{i=2, i\notin\{j,u\}}^{n} p_{ij}^{u} = y_{ju}$. Hence, (3.22a) essentially sets $y_{ju} = 1,\ \forall j \in PC_u,\ \forall u = 2,...,n$, which coincides with (3.20a) for the PCATSP2R^{-} formulation. Noting that (3.20b) and (3.20c) are identical with (3.22b) and (3.22c), respectively, we have that PCATSP-FL2 formulated as above is equivalent to PCATSP2R^{-}.

3.8 Computational Comparison of LP Relaxations of the Various PCATSP Formulations

Next, we present computational results to compare the LP relaxation bounds for the proposed PCATSP formulations, namely, PCATSP0, PCATSP1, PCATSP2, PCATSP2R, PCATSP2R^{-} ($\equiv$ PCATSP-FL2), PCATSP3, PCATSP4, PCATSP5, and PCATSP6 with those of six other models: PCATSP-SSB1, PCATSP-SD, PCATSP-GP, PCATSP-GP1, PCATSP-GP2, and PCATSP-GP3. Note that PCATSP-SSB1 has been shown in [25] to provide tighter lower bounds than PCATSP-SSB and PCATSP-SSB2, and that PCATSP4 is a further augmentation of PCATSP-SSB1 with the logical relationships (3.1f,g).

For the purpose of this comparison, we solved six standard test problems from the TSP library (TSPLIB), namely, esc07, esc11, esc12, br17.10, esc25 and esc47. The settings used in making the computational runs are the same as those used in Section 3.4. The results shown in Table 3-7 include the lower bound value (z_{LP}) obtained and the cpu time (in seconds) required to solve the LP relaxation of the different models. Note that the number in bold indicates the highest z_{LP} value obtained among all the models for each test case.

Table 3-7 reveals that PCATSP5 and PCATSP6 again provide the tightest lower bounds (z_{LP}), and that the lower bound values generated by PCATSP3 are also close to those obtained by PCATSP5 and PCATSP6. The LP-IP gap is typically less than 1.9%,

except for Problem br17.10 where the gap was 19.2%. Note that all these models contain the lifted RLT constraints (3.8a,b,c) along with the three-city subtour constraints (3.2) or its lifted version (3.9a). Also observe that unlike the corresponding runs for ATSP, PCATSP2R generates strictly tighter lower bounds than does PCATSP2R$^-$. This exhibits that the inclusion of (3.17) in (PC)ATSP2R strictly tightens the LP relaxation over that of (PC)ATSP2R$^-$ (or equivalently, (PC)ATSP-FL2).

Table 3-7 Results pertaining to LP relaxations of the various PCATSP formulations

Problem	Formulation	z_{LP}	CPU time	Problem	Formulation	z_{LP}	CPU time
esc07 (z_{IP}=2125)	PCATSP-SSB1	2125.00	0.01	**esc11** (z_{IP}=2075)	PCATSP-SSB1	2058.83	0.02
	PCATSP-SD	2098.21	0.01		PCATSP-SD	2041.41	0.01
	PCATSP-GP	1937.50	0.01		PCATSP-GP	2030.75	0.02
	PCATSP-GP1	2012.50	0.01		PCATSP-GP1	2030.75	0.02
	PCATSP-GP2	2060.00	0.01		PCATSP-GP2	2030.75	0.02
	PCATSP-GP3	2116.67	0.01		PCATSP-GP3	2030.75	0.03
	PCATSP0	**2125.00**	0.01		PCATSP0	2053.50	0.03
	PCATSP1	**2125.00**	0.02		PCATSP1	**2075.00**	0.25
	PCATSP2	**2125.00**	0.02		PCATSP2	**2075.00**	0.14
	PCATSP2R	**2125.00**	0.01		PCATSP2R	2057.50	0.05
	PCATSP2R$^-$	2087.50	0.01		PCATSP2R$^-$	2040.50	0.05
	PCATSP3	**2125.00**	0.01		PCATSP3	**2075.00**	0.19
	PCATSP4	**2125.00**	0.01		PCATSP4	2061.50	0.03
	PCATSP5	**2125.00**	0.01		PCATSP5	**2075.00**	0.14
	PCATSP6	**2125.00**	0.01		PCATSP6	**2075.00**	0.17
esc12 (z_{IP}=1675)	PCATSP-SSB1	1554.00	0.03	**br17.10** (z_{IP}=55)	PCATSP-SSB1	27.29	0.34
	PCATSP-SD	1520.35	0.02		PCATSP-SD	23.64	0.11
	PCATSP-GP	1507.50	0.02		PCATSP-GP	18.00	0.06
	PCATSP-GP1	1518.50	0.02		PCATSP-GP1	18.00	0.05
	PCATSP-GP2	1528.50	0.03		PCATSP-GP2	28.00	0.08
	PCATSP-GP3	1535.63	0.11		PCATSP-GP3	28.00	0.47
	PCATSP0	1507.50	0.03		PCATSP0	22.33	0.13
	PCATSP1	1649.50	0.95		PCATSP1	39.70	12.23
	PCATSP2	1649.50	0.38		PCATSP2	39.70	5.17
	PCATSP2R	1649.50	0.06		PCATSP2R	39.70	1.56
	PCATSP2R$^-$	1612.00	0.09		PCATSP2R$^-$	39.00	1.31
	PCATSP3	1649.50	0.36		PCATSP3	40.15	8.06
	PCATSP4	1566.40	0.05		PCATSP4	31.39	0.36
	PCATSP5	**1675.00**	0.45		PCATSP5	**44.45**	12.77
	PCATSP6	**1675.00**	0.48		PCATSP6	**44.45**	13.67
esc25 (z_{IP}=1681)	PCATSP-SSB1	1610.96	3.41	**esc47** (z_{IP}=1288)	PCATSP-SSB1	1242.37	378.05
	PCATSP-SD	1552.89	0.27		PCATSP-SD	1209.16	1.69
	PCATSP-GP	1537.94	0.34		PCATSP-GP	1214.83	14.30
	PCATSP-GP1	1540.17	0.41		PCATSP-GP1	1214.83	95.05
	PCATSP-GP2	1543.33	0.28		PCATSP-GP2	1215.94	20.59
	PCATSP-GP3	1548.29	0.48		PCATSP-GP3	1216.00	159.17
	PCATSP0	1607.72	3.13		PCATSP0	1219.00	389.47
	PCATSP1	1649.11	376.32		PCATSP1	1252.45	633,913.00
	PCATSP2	1649.11	124.93		PCATSP2	1252.45	466,216.00
	PCATSP2R	1637.29	12.23		PCATSP2R	1252.29	802.21
	PCATSP2R$^-$	1549.12	9.92		PCATSP2R$^-$	1232.00	516.41
	PCATSP3	1661.44	259.79		PCATSP3	1253.54	137,350.00
	PCATSP4	1644.04	6.27		PCATSP4	1242.75	489.57
	PCATSP5	**1681.00**	109.83		PCATSP5	**1263.02**	104,406.00
	PCATSP6	**1681.00**	143.52		PCATSP6	**1263.02**	111,281.00

3.9 Comment on Finding the Best Integer Solutions

Even though ATSP5 and ATSP6, as well as PCATSP5 and PCATSP6, provide the tightest LP-based lower bounds for the ATSP and PCATSP problems, respectively, the cpu times required to solve the LP-relaxations of these formulations are too large to make them competitive for solving the problems to optimality, especially for the ATSP problems. Yet, as indicated by our experimentation, the benefit of the RLT-based lifting process, as afforded by constraints (3.8a,b,c), is evident through the formulations ATSP2, ATSP2R, and ATSP2R$^-$, all of which accommodate these restrictions within ATSP0 while relaxing (3.1f,g,h), where the latter two of these formulations also relax (3.1e), either stating this as an inequality or deleting it altogether. In fact, for the ATSP test instances br17, ftv33, ftv35, ftv38, p43, and ftv44, the best reported overall performance (with regard to total cpu time) was obtained for the formulations ATSP2R, ATSP7, ATSP2R, ATSP2R$^-$, ATSP2R$^-$ (found best integer solution), and ATSP-SD (with ATSP2R$^-$, as next best), respectively. For the precedence constrained problems, PCATSP2R gave the best results for esc12, PCATSP-SD for esc47, while PCATSP4 (which lifts PCATSP-SSB1 along with the logical constraints (3.1f,g)), gave the best results for the remaining instances. Please refer to Appendix I, II and III for the detailed computational results pertaining to the LP relaxation and the best IP solutions of the various ATSP and PCATSP formulations.

While the computational results demonstrate the efficacy of employing the proposed theoretical RLT and logical lifting ideas, yet it remains of practical interest to take due advantage of the tightest formulations (PC)ATSP5 and (PC)ATSP6 developed herein. The key requirement to accomplish this is to be able to solve the underlying LP relaxations more effectively. One approach, to that end, would be to solve these LP relaxations to (near-) optimality using deflected subgradient methods on Lagrangian dual formulations (see Lim and Sherali [45], for example). In Section 3.5, we have presented some preliminary results in that regard. We have attempted to solve the LP relaxation of ATSP6 to (near-) optimality by using both a subgradient algorithm with average direction strategy (SA_ADS) (see Sherali and Ulular [69]) and two other nondifferentiable optimization (NDO) methods on the Lagrangian dual formulation of the associate problem, namely, the variable target value method (VTVM) (see Sherali *et al*. [66]) and

the trust region target value method (TRTV) (see Lim and Sherali [46]). The preliminary results show that the near optimal values obtained by the VTVM on solving the canonical format are the closest to the target optimal values.

Another approach might be to derive a set of strong valid inequalities based on these tighter formulations through a suitable surrogation process, and use these within the more compact manageable formulations. In Section 3.6, we suggest such a procedure to derive a set of strong valid inequalities. Our computational results show that, when the dual optimal solution is available, the associated strong valid inequalities generated from our procedure can successfully lift the LP relaxation of ATSP2R^{-} to be as tight as that of ATSP5 or ATSP6.

Chapter 4 Hot Strip Rolling Scheduling and Multiple-Asymmetric Traveling Salesman Problems

In this chapter, we consider a hot strip rolling scheduling problem that is encountered in a steel rolling mill. It involves processing of steel strips, of different sizes, by rollers in a pre-specified order. The processing of hot steel strips in rolling mills has been recognized as a bottleneck stage of steel production (see [36,40,42]). Therefore, an effective scheduling of hot steel strips at this stage of production can help in enhancing overall productivity. The processing of these strips in rolling mills may involve the use of a single-roller or multiple rollers. The single-roller hot strip rolling problem has been formulated as an asymmetric traveling salesman problem in the literature (see [9] and [10]). However, no specific formulations have been presented in the literature for the multiple-roller problem. Here, we present several formulations for this problem that also capture a given special type of precedence order in which the hot steel strips are processed, and also, investigate the tightness of these models.

The organization of this chapter is as follows. In Section 4.1, we describe the hot strip rolling scheduling problem and present its characteristics. We review the formulations and algorithms that have been presented for this problem in the literature in Section 4.2. In Section 4.3, we first consider the single-roller hot strip rolling scheduling problem and use the PCATSP formulations presented in Chapter 3 for the solution of this problem. Then, we present several new formulations for the multiple-asymmetric traveling salesman problem. Subsequently, these are used to model the hot strip rolling scheduling problem involving multiple rollers. These formulations are also used to model the hot strip rolling scheduling problem with slab selection, for the case of single roller and multiple rollers. In Section 4.4, we design a series of computational experiments to depict the tightness of the LP relaxations of our formulations and to

explore the solution of one of these formulations to optimality using different mixed-integer programming algorithms.

4.1 Problem Description

The steel industry is one of the most competitive industries as is depicted by the fact that the ten biggest steel companies in the world have a combined market share of only 26.5 percent. According to the 2005 annual report published by the International Iron and Steel Institute (IISI), the average annual growth rate for steel doubled from 2.4%, over the period from 1995-2004, to 5.7%, over the period from 2000-2004 [36]. Furthermore, the IISI had estimated a similar trend for 2005, that is, a growth of 5.8% over 2004. As the third largest steel producer in the world, North America accounted for 11.2% of the world's total steel production. However, the steel production in the US decreased by 5.8% in 2005 to 93.9 million tons [37].

In order to highlight the importance of sustainability in steel industry, IISI has drawn up a set of 11 sustainability indicators. Four of these indicators pertain to economic criteria: investment in new processes and products, operating margins, return on capital employed, and value added [35]. Clearly, to achieve appropriate operating margins, profits must be high enough to justify investment in new products and processes, which in turn, is essential to ensure the survival and growth of steel industry. While the price of steel is subject to global competition, one sure way of increasing profit is by lowering the production cost incurred. And, a proper scheduling of work plays a key role in reducing production cost by reducing waste and maintaining a high production rate.

Here, we consider the hot strip rolling scheduling problem that is encountered in a steel rolling mill. The hot strip rolling contributed to about 17% of the world steel exports in 2003 [36]. Therefore, any enhancement in the performance of production schedules used in the steel rolling mills would result in significant overall benefits.

According to Lee and Murthy [42], the hot strip rolling scheduling problem that arises in a steel plant is one of the most difficult industrial scheduling problems due to the intricacies involved in synchronizing the complex and separate processes. The primary operations performed at a steel rolling mill are shown in Figure 4-1. The production of

steel involves two major steps: primary and finishing. First, liquid iron is made in a batch by putting raw material such as iron ore and coal into a blast furnace. Each batch of liquid iron produces about 16 long bars, or slabs, with identical chemical composition. Depending upon the status of demand, the slabs are either stored in a local yard or are sent directly to a finishing mill.

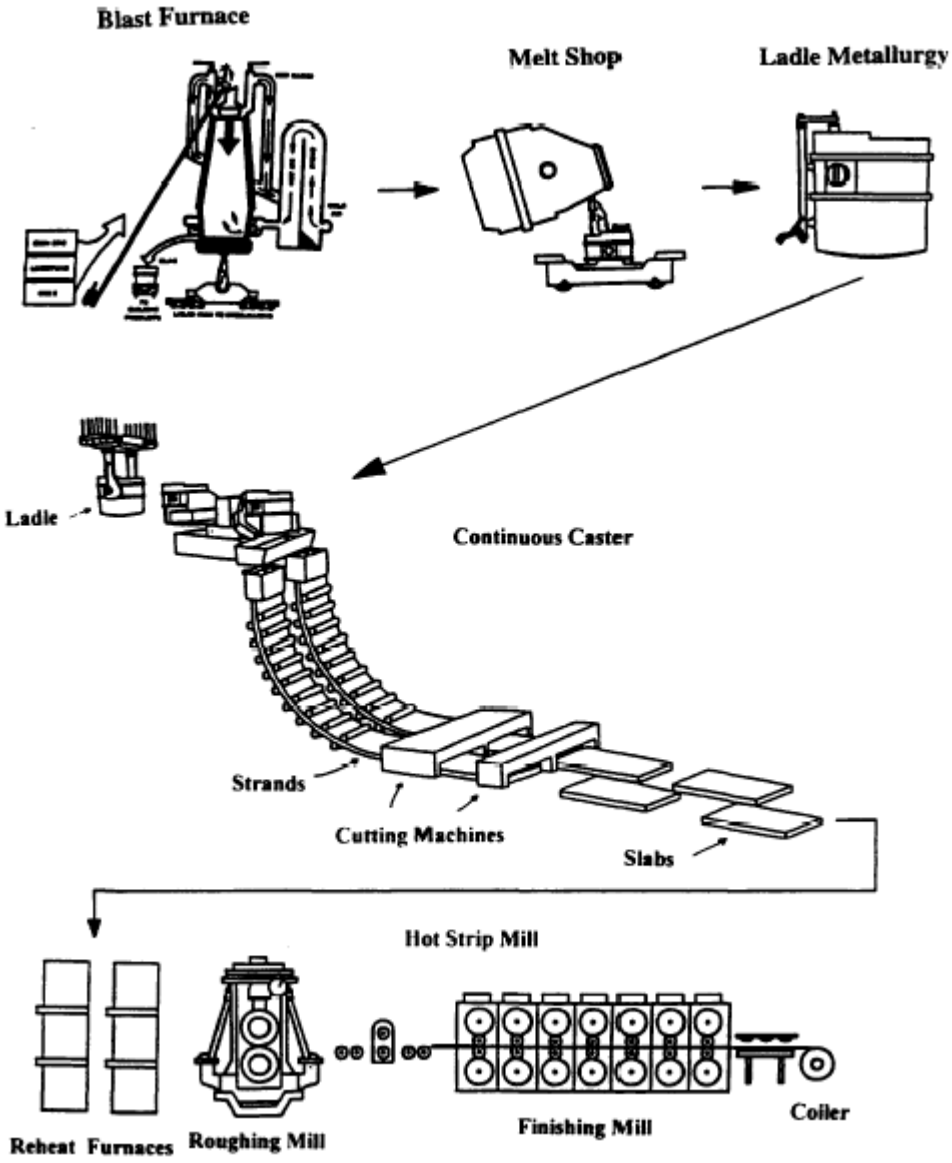

Figure 4-1 The primary operations performed at a steel rolling mill [47]

A finishing mill, or a hot strip rolling mill, transforms slabs into long rolls, or steel coils. Before rolling, the slabs go through a reheat furnace in order to heat the slabs to a desired temperature, called the drop-out temperature. After heating, the thickness of the

slabs is reduced gradually by running them through a set of rollers. Since the rollers undergo wear and tear as the slabs pass through them, they have to be replaced after a certain number of slabs has been processed. The set of slabs that are processed between the two consecutive changes of a roller is called a turn or a product block. For a detailed description of the hot rolling procedure, please see [40,47,72].

Each slab that is processed through the rollers has the following characteristics: width, thickness, hardness, gauge, chemical composition, charging temperature, and drop-out temperature. The charging temperature is the temperature of a slab right before it enters a reheat furnace, and together with its drop-out temperature, it determines the processing time of the slab in a reheat furnace. In order to reduce the waste of energy in the reheat area, it is desired to achieve a smooth transition of processing times between two consecutive slabs in a reheat furnace. A characteristic that is of paramount importance for the sequencing of slabs in the rollers is their width.

Figure 4-2 depicts the desired profile of the widths of the slabs that are to be processed through the rollers. This profile avoids formation of marks on the slabs. As noted earlier, the set of slabs that is produced between two roller setups is called a turn (see [40,72]). There are two parts in a turn: a warm-up part and a coming-down part. The warm-up part is of shorter length and includes the break-in and wide-out slabs, from narrow to wide, while the coming-down part is a major part of a turn and consists of slabs of widths from wide to narrow.

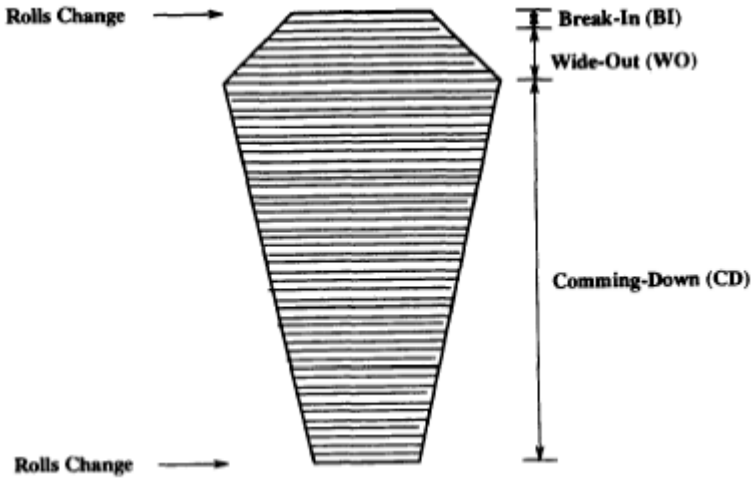

Figure 4-2 The desired profile of the widths of the slabs in a turn [47]

Smooth changes in hardness and gauges are necessary while sequencing the slabs in a turn. The wear and tear of the rollers depend on the forces that are exerted on the rolling mill by the slabs being processed. In other words, the lower the overall changes in hardness and gauge from one slab to the next, the larger the number of slabs that can be processed in a turn. The other characteristics of the slabs, like hardness and gauge, impact the wear and tear of a roller. A smoother transition among the slabs with respect to their hardness and gauge causes less wear and tear. As a result, a greater number of slabs could be added in a turn under smoother transition. For our analysis, we assume a given number of slabs that is to be processed in a turn. The hot strip rolling scheduling problem can, thus, be stated as follows: Given a set of slabs of different widths, determine the slabs to be processed in each turn so that the desired profile of the widths of the slabs in each turn is maintained and sequence-dependent setup cost, measured in terms of roller wear caused by the processing of the slabs of different characteristics, is minimized.

4.2 Literature Review

Kosiba *et al.* [40] formulate the hot strip rolling scheduling problem as a traveling salesman problem with asymmetric cost matrices. A penalty structure, which is created by the scheduling experts at the Bethlehem Steel Corporation's Burns Harbor Mill, is used to indicate the expected damage to rollers caused by changes in the width, hardness, and gauge of two consecutive slabs. The objective of this ATSP is to minimize the total penalty incurred. They group identical slabs to further reduce the problem size and apply the algorithm by Miller and Pekny [52] to obtain an optimal schedule. In their computational result, they use four datasets and report improvements, ranging from 44% to 78%, in total penalty due to the use of optimal schedules over the one used in practice. Their algorithm is based on the branch and bound approach. It also uses a specific interchange heuristic to obtain an upper bound value. First, at each node, the cost matrix is modified as follows.

$$c'_{ij} = \begin{cases} c_{ij}, \text{ if } c_{ij} \leq \lambda, \\ \infty, \text{ otherwise,} \end{cases}$$

where c_{ij} is the penalty of processing slab j after slab i, and λ is a threshold value to ensure that only penalty cost lower than this value is considered in the algorithm. This modified cost matrix is sparser and is easier to use to obtain an optimal solution. The optimal solution is identified by investigating the difference between the optimal value of this modified ATSP, $val(ATSP')$, and a lower bound, $val(AP)$, of the original ATSP obtained by including only the assignment constraints. It is shown that if we let u' and v' be the dual variables corresponding to the assignment constraints for the modified ATSP, then, an optimal solution x' for the modified ATSP is also optimal for the original ATSP if $val(ATSP') - val(AP) \le \lambda + 1 - u'_i - v'_{\max}$ for all $i \in N$, and $\lambda + 1 - u'_i - v'_{\max} \ge 0$.

Balas [8] considers the hot strip rolling scheduling problem for the case of selecting a set of slabs from an inventory of available slabs. He presents a formulation for this problem, termed the prize collecting traveling salesman problem, as a generalization of the traveling salesman problem. The problem can be stated as follows: A traveling salesman gets a prize w_i for visiting city i, pays a penalty p_i for skipping city i, and a traveling cost c_{ij} for moving from city i to city j in his tour. The problem is to find the best tour for the traveling salesman that minimizes the sum of traveling costs and net penalties, while still collecting at least w_0 in total prizes from the cities that he visits. Let $y_i = 1$, if city i is visited on the tour; and 0, otherwise; and $x_{ij} = 1$, if city j is visited immediately after city i; and 0, otherwise. The prize collecting TSP is formulated on a complete graph $G' = (N, A)$ as follows.

$$\text{Minimize} \sum_{i=1}^{n} \sum_{j=1, j\neq i}^{n} c_{ij} x_{ij} + \sum_{i=1}^{n} p_i (i - y_i) \tag{4.1a}$$

subject to

$$\sum_{i=1, i\neq j}^{n} x_{ij} - y_j = 0 \ , \ \forall j = 1,...,n \tag{4.1b}$$

$$\sum_{j=1, j\neq i}^{n} x_{ij} - y_i = 0 \ , \ \forall i = 1,...,n \tag{4.1c}$$

$$\sum_{i=1}^{n} w_i y_i \ge w_0 \tag{4.1d}$$

$$x_{ij} \in \{0,1\}\ ,\ \forall i, j = 1,...,n\ ,\ i \neq j\ ,\ y_i \in \{0,1\}\ ,\ \forall i = 1,...,n \tag{4.1e}$$

$$G'(y,x) \text{ is a cycle.} \tag{4.1f}$$

Note $G'(y,x)$ is a subgraph of G' whose nodes and arcs are those that are defined by $\boldsymbol{y}$ and $\boldsymbol{x}$, respectively. Furthermore, the variables $y_i,\ \forall i = 1,...,n$, can be complemented by introducing n new variables $x_{ii} = 1 - y_i,\ \forall i = 1,...,n$, which are interpreted as the arcs whose head and tail are identical. Let $c_{ii} = p_i$ for city $i,\ \forall i = 1,...,n$, and $U = \sum_{i=1}^{n} w_i - w_0$. Then, the formulation in (4.1) can be rewritten as follows.

$$\text{Minimize } \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij} x_{ij} \tag{4.2a}$$

subject to

$$\sum_{i=1}^{n} x_{ij} = 1\ ,\ \forall j = 1,...,n \tag{4.2b}$$

$$\sum_{j=1}^{n} x_{ij} = 1\ ,\ \forall i = 1,...,n \tag{4.2c}$$

$$\sum_{i=1}^{n} w_i x_{ii} \leq U \tag{4.2d}$$

$$x_{ij} \in \{0,1\}\ ,\ \forall i, j = 1,...,n \tag{4.2e}$$

$$G(x) \text{ has exactly one cycle of length} \geq 2\ . \tag{4.2f}$$

A typical solution to this formulation on 5 nodes is shown in Figure 4-3.

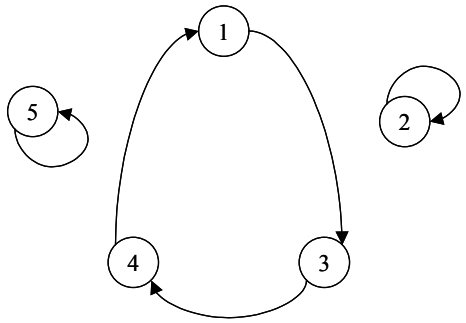

Figure 4-3 A typical solution for prize collecting TSP on 5 nodes [8]

In this paper, Balas [8] focuses on studying the structural properties of the prize collecting TSP and provides no algorithms for its solution. Besides, the formulations in (4.1) and (4.2) are conceptual models rather than compact ones.

Lopez *et al.* [47] consider the hot strip rolling scheduling and also model it as a prize collecting TSP. They try to obtain a good schedule for processing the slabs that minimizes the changes experienced in both the reheat and rolling mill areas. These changes are due to variations in width, hardness, gauge, and residence time in the reheat furnace between two consecutive slabs of a schedule. They propose a heuristic based on Tabu search to determine a near optimal solution. In their computational experiment, they test the proposed heuristic on datasets obtained from Dofasco in Hamilton, Ontario, Canada. Based on the computational results, they have reported that the solutions generated by their heuristic outperforms the schedules used in practice both in terms of increasing the number of slabs scheduled in a turn and in lowering the total penalty encountered.

Assaf *et al.* [4] have proposed a branch and bound-based method to generate optimal schedules for steel production. They argue that a large number of constraints effectively reduce the number of feasible schedules. Consequently, an optimal schedule can be obtained within an acceptable cpu time. Their algorithm includes the following three processes: branching and pruning, updating upper bound, and cost evaluation. In the branching and pruning process, they enforce seven constraints in order to generate a feasible schedule and have applied these constraints to prune the branching tree. Also, they use simulation results of four sub-models to evaluate the total cost for the remaining branches, update upper bounds and to help in fathoming the branching tree. Their solution, however, does not always guarantee savings over the schedule used by IPSCO Inc., Canada.

Tang *et al.* [72] have developed a model and solution method for the hot strip rolling scheduling problem of the Shanghai Baoshan Iron and Steel Complex. They consider the scheduling of the rolling production operation for a production shift, which consists of 5 to 7 turns (see Figure 4-4). The objective of their scheduling problem is to minimize the total penalty cost, which is identical to the one used by Kosiba *et al.* [40]. Although they

allude to the hot strip rolling scheduling problem as a multiple-traveling salesman problem (mTSP), yet they resort to using a single-traveling salesman problem (TSP) model. They provide a modified genetic algorithm to obtain a near optimal solution and report improvements in its implementation over the manual system used in the facility.

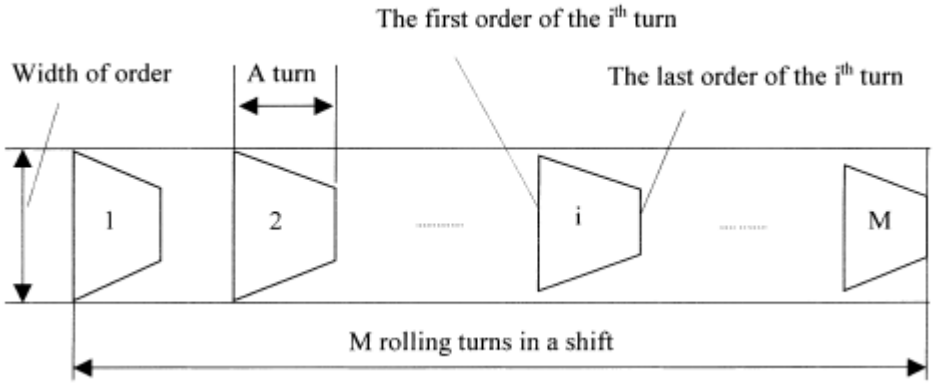

Figure 4-4 A hot rolling schedule for a shift [72]

Petersen *et al.* [58] use dynamic programming to sequence slabs through the reheat furnace and rolling mill. The objective function that they consider is an aggregated function of the following three criteria: minimizing deviation from the synchronized production, minimizing wasted heating area, and minimizing delivery time. They propose a modified greedy approach to construct a decision tree and determine an optimal sequence by total enumeration, and report on its implementation in a Danish steel mill.

Appelqvist and Lehtonen [2] have presented a scheduling package, which has been used as a combined approach of optimization and discrete-event simulation, for a steel mill in Finland. They use the optimization algorithm proposed by Assaf *et al.* [4] to generate initial schedules, and then, verify these schedules by a simulation model. Their implementation results have shown an increment in production volumes. Also, they report that the simulation model almost always verified the schedules that were used after two years of system implementation.

4.3 New Formulations for the Hot Strip Rolling Scheduling Problem

The hot strip rolling scheduling problem is concerned with the determination of a sequence in which to process a given set of slabs so that the total penalty is minimized. We consider four types of hot strip rolling scheduling problems that may arise in practice. First, we consider the sequencing of a given set of slabs for one turn only. We call this problem as a single-roller hot strip rolling scheduling problem, or simple a single-roller problem, which can be formulated as a precedence-constrained asymmetric traveling salesman problem (PCATSP). Then, we consider the scheduling of a given set of slabs for simultaneous processing on more than one turn. We designate this problem as a multiple-roller hot strip rolling scheduling problem, or a multiple-roller problem in short. The third problem that we consider has to do with the selection of slabs from a given set of slabs, but still for a single turn. Thus, in this case, not all the slabs need to be processed in one turn. We designate this problem as a single-roller hot strip rolling scheduling problem with slab selection, and we refer to it as a single-roller problem with slab selection. Our last problem involves the selection of slabs from a given set of slabs for simultaneous processing on multiple turns. Accordingly, we call this as a multiple-roller hot strip rolling scheduling problem with slab selection, or a multiple-roller problem with slab selection in short.

As noted above, the first of these four problems is identical to the ATSP but involves precedence constraints of the type shown in Figure 4-2. However, we model each of the remaining three problems as a variation of the underlying multiple-asymmetric traveling salesman problem (mATSP) to capture the stated requirements for each. Since the mATSP is at the core of each of these three problems, in the sequel, we first address the basic mATSP. We present several new formulations for this problem and also present results depicting the tightness of these formulations. Subsequently, this formulation is adapted to model the multiple-roller problem, the single-roller scheduling problem with slab selection and the multiple-roller scheduling problem with slab selection.

4.3.1 Single-roller scheduling problem

In view of the order in which the slabs are to be processed, we can formulate this problem as a precedence-constrained traveling salesman problem. Suppose that there are n slabs to be rolled in one turn. Without lost of generality, we include a dummy slab, designated as $\{0\}$, which behaves as the first and the last slab in a turn. Let $c_{ij} = p_{ij}^{w} + p_{ij}^{g} + p_{ij}^{h}$ be the setup cost if a slab j is rolled right after slab i, where p_{ij}^{w}, p_{ij}^{g}, and p_{ij}^{h} represent the penalties as a result of changes in the width, gauge, and hardness from slab i to slab j, $\forall i, j = 1,...,n$, $i \neq j$, respectively. The precedence relationship in which the slabs are to be processed is given in Figure 4-2. Let PC_j be the set of slabs that are required to precede slab j, $\forall j = 1,...,n$, and we denote this as $i \prec j$ if $i \in PC_j$. As a result, all PCATSP formulations in Chapter 3 are valid for this problem. Next, we make an additional observation regarding the tightness of the polytope for PCATSP.

Balas *et al.* [10] define two facets of the PCATSP polytope, $conv(\text{PCATSP})$, as follows. Assume that node 1 is the base node. Let $n \geq 5$ and v be a free node, which has no precedence relationships with other nodes. Let $\hat{S}$ be a subset of nodes excluding the base node. Define the predecessors and successors of $\hat{S}$ as $\pi(\hat{S}) = \{i \in \{1,...,n\} : \exists j \ni i \prec j, j \in \hat{S}\}$ and $\sigma(\hat{S}) = \{k \in \{1,...,n\} : \exists j \ni j \prec k, j \in \hat{S}\}$, respectively. Moreover, we define the π-inequalities and σ-inequalities as follows. For any $S \subseteq \{1,...,n\} \setminus \{1, v\}$ such that $\pi(S) \subset S$ and $\sigma(S) \subset S$, the π-inequalities: $x(S \setminus \pi(S), \overline{S}) \geq 1$, and the σ-inequalities: $x(\overline{S}, S \setminus \sigma(S)) \geq 1$, define facets of $conv(\text{PCATSP})$.

Proposition 4.1. PCATSP2 includes the π-inequalities and σ-inequalities.

Proof. We prove this result by contradiction. Assume that there exists a feasible solution for PCATSP2 that violates the π-inequalities; i.e., $\sum_{i \in S \setminus \pi(S)} \sum_{j \in \overline{S}} x_{ij} \leq 0$. Since $\pi(S) \cap \overline{S} = \phi$, the inequalities can also be written as $\sum_{i \in S} \sum_{j \in \overline{S}} x_{ij} - \sum_{i \in \pi(S)} \sum_{j \in \overline{S}} x_{ij} \leq 0$, which

implies that $\sum_{i\in S}\sum_{j\in \bar{S}} x_{ij} = \sum_{i\in \pi(S)}\sum_{j\in \bar{S}} x_{ij}$ (or $\sum_{i\in S\setminus\pi(S)}\sum_{j\in \bar{S}} x_{ij} = 0$) holds due to the fact that $\pi(S)\subset S$. Therefore, for every node in $S\setminus\pi(S)$, it can either generate a flow to the set $\bar{S}$ by going through some nodes in $\pi(S)$ or there is no flow to the set $\bar{S}$. From constraints (3.8b) and (3.20a), a feasible solution to PCATSP2 includes a path from node k to node i with total flow of 1 for $k\in\pi(S)$ and $k\prec i$; i.e. $\sum_{j\in\pi(S)} x_{ji} = 1$, $\forall i\in S\setminus\pi(S)$. Furthermore, for those nodes $i\in S\setminus\pi(S)$, the feasible solution requires exactly one unit of flow sent back to the base node $\{1\}$, or the set $\bar{S}$; i.e., $\sum_{j\in\bar{S}} x_{ij} = 1$, which implies that $\sum_{j\in\pi(S)} x_{ij} = 1$, $\forall i\in S\setminus\pi(S)$. This results in a circuit between set $\pi(S)$ and node i, $\forall i\in S\setminus\pi(S)$, which contradicts that this solution is feasible to PCATSP2.

Similarly, let us assume that there exists some feasible solution for PCATSP2 that violates the σ-inequalities; i.e., $\sum_{i\in\bar{S}}\sum_{j\in S\setminus\sigma(S)} x_{ij} \le 0$. Since $\bar{S}\cap\sigma(S)=\phi$, the inequalities can also be written as $\sum_{i\in\bar{S}}\sum_{j\in S} x_{ij} - \sum_{i\in\bar{S}}\sum_{j\in\sigma(S)} x_{ij} \le 0$. Since $\sigma(S)\subset S$, we have either $\sum_{i\in\bar{S}}\sum_{j\in S} x_{ij} = \sum_{i\in\bar{S}}\sum_{j\in\sigma(S)} x_{ij}$ or $\sum_{i\in\bar{S}}\sum_{j\in S\setminus\sigma(S)} x_{ij} = 0$. Furthermore, there must exist exactly one unit of incoming flow from the set $\bar{S}$ to every node in $S\setminus\sigma(S)$ for a feasible solution to PCATSP2 due to the fact that the base node $\{1\}$ is in the set $\bar{S}$; i.e., $\sum_{j\in\bar{S}} x_{ji} = 1$, $\forall i\in S\setminus\sigma(S)$. This can only hold when $\sum_{j\in\sigma(S)} x_{ji} = 1$, $\forall i\in S\setminus\sigma(S)$. Also, from constraints (3.8b) and (3.20a), there exists a path from node k to node i with a total flow of 1 for $k\in\sigma(S)$ and $i\prec k$; i.e. $\sum_{j\in\sigma(S)} x_{ij} = 1$, $\forall i\in S\setminus\sigma(S)$. This results in a circuit between node k and node i, which contradicts that this solution is feasible to PCATSP2. This completes the proof. □

From Proposition 4.1, we can conclude that PCATSP2, and tighter formulations like PCATSP3 and PCATSP6, give a better LP-relaxation than the one that includes the π-

and σ-inequalities given in Balas *et al.* [10]. Moreover, the tightness of these inequalities as facet defining can only be ensured if there exists at least one free node, with no precedence relationships. However, this restricted condition might not hold in practice as is evident from the single hot strip rolling scheduling problem on hand.

4.3.2 Multiple-asymmetric traveling salesman problem

The multiple-asymmetric traveling salesman problem (mATSP) is a generalization of the single-asymmetric traveling salesman problem (ATSP), and it can be defined as follows: Given a set of cities and m traveling salesmen, determine m tours, one for each salesman, such that, starting from the same base city, each salesman visits a subset of cities and returns to the base city; with each city to be visited by only one salesman, in order to minimize the total distance traveled by all salesmen.

We have reviewed several formulations presented in the literature for this problem, including mATSP-MTZ and mATSP-Bektas, in Section 2.2. Next, we present some new formulations for the mATSP and later adapt these to the hot strip rolling scheduling problem on hand.

4.3.2.1 New formulations for the multiple-asymmetric traveling salesman problem (mATSP)

Without loss of generality, we assume that all salesmen begin their tours from the same city, namely, city 1. We require the following variables for our formulation.

$$x_{ij}^{t} = \begin{cases} 1 \text{ , if city } i \text{ directly precedes city } j \text{ on the tour of salesman } t, \\ 0 \text{ , otherwise, } \forall i, j = 1,...,n,\ i \neq j,\ \forall t = 1,...,m, \end{cases}$$

$$y_{ij}^{t} = \begin{cases} 1 \text{ , if city } i \text{ precedes city } j \text{ on the tour of salesman } t, \\ 0 \text{ , otherwise, } \forall i, j = 2,...,n,\ i \neq j,\ \forall t = 1,...,m, \end{cases}$$

and

$$z_{i}^{t} = \begin{cases} 1 \text{ , if city } i \text{ is visited by salesman } t, \\ 0 \text{ , otherwise, } \forall j = 2,...,n,\ \forall t = 1,...,m. \end{cases}$$

Note that the $\boldsymbol{x}$-variables define a Hamiltonian circuit for each salesman, whereas the $\boldsymbol{y}$-variables view the mATSP solution as Hamiltonian paths that start from the base city to other cities. The $\boldsymbol{z}$-variables determine the cities visited by each salesman.

mATSP0: Minimize $\sum_{i=1}^{n}\sum_{j=1,j\neq i}^{n} c_{ij}\cdot\left(\sum_{t=1}^{m} x_{ij}^{t}\right)$ (4.3a)

subject to

$$\sum_{i=1,i\neq j}^{n} x_{ij}^{t} = z_{j}^{t}\ ,\ \forall j=2,...,n\ ,\ \forall t=1,...,m \tag{4.3b}$$

$$\sum_{j=1,j\neq i}^{n} x_{ij}^{t} = z_{i}^{t}\ ,\ \forall i=2,...,n\ ,\ \forall t=1,...,m \tag{4.3c}$$

$$\sum_{t=1}^{m} z_{i}^{t} = 1\ ,\ \forall i=2,...,n \tag{4.3d}$$

$$\sum_{i=2}^{n} x_{i1}^{t} = 1\ ,\ \forall t=1,...,m \tag{4.3e}$$

$$\sum_{j=2}^{n} x_{1j}^{t} = 1\ ,\ \forall t=1,...,m \tag{4.3f}$$

$$x_{1j}^{t} \le z_{j}^{t}\ ,\ \forall j=2,...,n\ ,\ \forall t=1,...,m \tag{4.3g}$$

$$x_{i1}^{t} \le z_{i}^{t}\ ,\ \forall i=2,...,n\ ,\ \forall t=1,...,m \tag{4.3h}$$

$$x_{ij}^{t} \le y_{ij}^{t}\ ,\ \forall i,j=2,...,n\ ,\ i\neq j\ ,\ \forall t=1,...,m \tag{4.3i}$$

$$y_{ij}^{t} \le z_{i}^{t}\ ,\ \forall i,j=2,...,n\ ,\ i\neq j\ ,\ \forall t=1,...,m \tag{4.3j}$$

$$y_{ij}^{t} \le z_{j}^{t}\ ,\ \forall i,j=2,...,n\ ,\ i\neq j\ ,\ \forall t=1,...,m \tag{4.3k}$$

$$y_{ij}^{t} + y_{ji}^{t} \le 1\ ,\ \forall i,j=2,...,n\ ,\ i\neq j\ ,\ \forall t=1,...,m \tag{4.3ℓ}$$

$$y_{ij}^{t} + y_{ji}^{t} \ge z_{i}^{t} + z_{j}^{t} - 1\ ,\ \forall i,j=2,...,n\ ,\ i\neq j\ ,\ \forall t=1,...,m \tag{4.3m}$$

$$-\left(1-x_{iv}^{t}\right) \le \left(y_{ij}^{t} - y_{vj}^{t}\right) \le \left(1-x_{iv}^{t}\right),\ \forall i,j,v=2,...,n\ ,\ i\neq j\neq v\ ,\ \forall t=1,...,m \tag{4.3n}$$

$$x_{ij}^t = \{0,1\},\ \forall i, j = 1,...,n,\ i \neq j,\ \forall t = 1,...,m \tag{4.3o}$$

$$z_i^t \geq 0,\ \forall i = 2,...,n,\ \forall t = 1,...,m;\ y_{ij}^t \geq 0,\ \forall i, j = 2,...,n,\ i \neq j,\ \forall t = 1,...,m. \tag{4.3p}$$

In mATSP0, (4.3b-d,o,p) are assignment constraints for cities other than the base city; (4.3e,f,o) are assignment constraints that enforce that every tour starts from and ends at the base city; (4.3g,h) assert that a salesman can only depart to and arrive from a city which is assigned to that salesman; (4.3i,ℓ) logically relate the precedence variables $\boldsymbol{y}$ to the immediate precedence variables $\boldsymbol{x}$; (4.3j,k) capture the fact that two cities i and j should be related by precedence only if they are assigned to the same salesman; (4.3m) along with (4.3ℓ) assert that either city i should precede city j or vice versa, $\forall i, j = 2,...,n,\ i \neq j$, if they are assigned to the same salesman t, $\forall t = 1,...,m$, and (4.3n) enforces that if $x_{iv}^t = 1$ for any $i \neq v$ on a tour t, then for any other city j, we must have $y_{ij}^t = y_{vj}^t$, whether this pair equals 0 or 1.

The above formulation is an adaptation of the formulation for the ATSP, presented in Chapter 3, to the case of m traveling salesmen. Consequently, the validity of mATSP0 trivially follows from the arguments presented in Chapter 3.

Proceeding in a manner similar to that presented in Chapter 3, we can tighten the formulation mATSP0 by applying the first-order RLT process of Sherali and Adams [63,64,65]. Consider the following special RLT product constraint using (4.3b), (4.3c), and the implied constraint $0 \leq y_{vj}^t \leq 1,\ \forall i, j = 2,...,n,\ i \neq j,\ \forall t = 1,...,m$ (from (4.3p)):

$$\text{(i)}\quad \left(0 \leq y_{vj}^t \leq 1\right) * x_{iv}^t,\ \forall i = 1,...,n,\ \forall v, j = 2,...,n\ \ i \neq j \neq v,\ \forall t = 1,...,m. \tag{4.4a}$$

$$\text{(ii)}\quad \left[\sum_{v=1, v\neq i}^{n} x_{iv}^t = z_i^t\right] * y_{ij}^t,\ \forall i, j = 2,...,n,\ i \neq j,\ \forall t = 1,...,m. \tag{4.4b}$$

$$\text{(iii)}\quad \left[\sum_{i=1, i\neq v}^{n} x_{iv}^t = z_v^t\right] * y_{vj}^t,\ \forall v, j = 2,...,n,\ v \neq j,\ \forall t = 1,...,m. \tag{4.4c}$$

To linearize the product terms thus created, let

$$(f_{ij}^v)^t = x_{iv}^t \cdot y_{vj}^t,\ \forall i = 1,...,n,\ \forall v, j = 2,...,n\ \ i \neq j \neq v,\ \forall t = 1,...,m. \tag{4.5}$$

It is easy to show (see Section 3.2) that the constraints of mATSP0 imply the following identities.

$$x_{iv}^t \cdot y_{ij}^t = (f_{ij}^v)^t,\ \forall i = 1,...,n,\ \forall v, j = 2,...,n\ \ i \neq j \neq v,\ \forall t = 1,...,m. \tag{4.6a}$$

$$x_{iv}^t \cdot y_{iv}^t = x_{iv}^t,\ \forall i, v = 2,...,n,\ i \neq v,\ \forall t = 1,...,m. \tag{4.6b}$$

$$x_{iv}^t \cdot y_{vi}^t = 0,\ \forall i, v = 2,...,n,\ i \neq v,\ \forall t = 1,...,m. \tag{4.6c}$$

$$x_{i1}^t \cdot y_{ij}^t = 0,\ \forall i, j = 2,...,n,\ i \neq j,\ \forall t = 1,...,m. \tag{4.6d}$$

$$z_i^t \cdot y_{ij}^t = y_{ij}^t,\ \forall i, j = 2,...,n,\ i \neq j,\ \forall t = 1,...,m. \tag{4.6e}$$

These identities, (4.6a) – (4.6e), along with (4.5), can be used in RLT product constraints (4.4a,b,c) to obtain the following respective linearized constraints.

$$0 \leq (f_{ij}^v)^t \leq x_{iv}^t,\ \forall i = 1,...,n,\ \forall v, j = 2,...,n\ \ i \neq j \neq v,\ \forall t = 1,...,m. \tag{4.7a}$$

$$\sum_{v=2, v \notin \{i,j\}}^{n} (f_{ij}^v)^t + x_{ij}^t = y_{ij}^t,\ \forall i, j = 2,...,n,\ i \neq j,\ \forall t = 1,...,m. \tag{4.7b}$$

$$\sum_{i=1, i \notin \{v,j\}}^{n} (f_{ij}^v)^t = y_{vj}^t, \forall v, j = 2,...,n,\ v \neq j,\ \forall t = 1,...,m. \tag{4.7c}$$

Note that (4.7a) is obtained directly by substituting (4.5) in (4.4a), (4.7b) is obtained by using (4.6a,b,d,e) in (4.4b), and (4.7c) is obtained by using (4.5) along with (4.6c,e) in (4.4c). These constraints can now be augmented to mATSP0 to obtain the following tightened RLT-lifted reformulation of mATSP0. Moreover, note that (4.3i) can be omitted because it is implied by (4.7b).

mATSP1: Minimize $\left\{ \text{(4.3a): (4.3b)-(4.3h),(4.3j)-(4.3p), and (4.7a)-(4.7c)} \right\}$.

In our analysis below, we shall also consider the following relaxation of mATSP1 obtained by deleting (4.3n):

mATSP2: Minimize $\left\{ \text{(4.3a): (4.3b)-(4.3h),(4.3j)-(4.3m), (4.3o,p), and (4.7a)-(4.7c)} \right\}$.

To establish the validity of mATSP2, we consider the maximum flow ATSP formulation of Wong [75], and extend it to the case of m traveling salesmen by enforcing a maximum flow of at least a unit from the source node to all other nodes for every route

that starts and ends at the base city. Let mATSP-DFJ be the formulation of mATSP with exponential number of Dantzig-Fulkerson-Johnson (DFJ) [19] subtour elimination constraints for each salesman. Since Wong [75] has shown that the maximum flow formulation for ATSP is equivalent to the ATSP formulation with DFJ subtour elimination constraints, this equivalence holds for the m traveling salesmen versions of their formulations as well.

The maximum flow formulation for the mATSP, referred to as mATSP-FL, is as follows:

mATSP-FL: Minimize $\sum_{i=1}^{n} \sum_{j=1, j\neq i}^{n} c_{ij} \cdot \left(\sum_{t=1}^{m} x_{ij}^{t} \right)$ (4.8a)

subject to

$$\sum_{i=1, i\neq j}^{n} x_{ij}^{t} = z_{j}^{t} \ , \ \forall j = 2,...,n \ , \ \forall t = 1,...,m \tag{4.8b}$$

$$\sum_{j=1, j\neq i}^{n} x_{ij}^{t} = z_{i}^{t} \ , \ \forall i = 2,...,n \ , \ \forall t = 1,...,m \tag{4.8c}$$

$$\sum_{t=1}^{m} z_{i}^{t} = 1 \ , \ \forall i = 2,...,n \tag{4.8d}$$

$$\sum_{i=2}^{n} x_{i1}^{t} = 1 \ , \ \forall t = 1,...,m \tag{4.8e}$$

$$\sum_{j=2}^{n} x_{1j}^{t} = 1 \ , \ \forall t = 1,...,m \tag{4.8f}$$

$$x_{1j}^{t} \leq z_{j}^{t} \ , \ \forall j = 2,...,n \ , \ \forall t = 1,...,m \tag{4.8g}$$

$$x_{i1}^{t} \leq z_{i}^{t} \ , \ \forall i = 2,...,n \ , \ \forall t = 1,...,m \tag{4.8h}$$

$$(p_{ij}^{u})^{t} \leq x_{ij}^{t} \ , \ \forall u = 2,...,n \ , \ \forall i, j = 1,...,n \ , \ i \neq j \ , \ \forall t = 1,...,m \tag{4.8i}$$

$$\sum_{j=2}^{n} (p_{1j}^{u})^{t} - \sum_{j=2}^{n} (p_{j1}^{u})^{t} = z_{u}^{t} \ , \ \forall u = 2,...,n \ , \ \forall t = 1,...,m \tag{4.8j}$$

$$\sum_{j=1, j\neq i}^{n} (p_{ij}^{u})^{t} - \sum_{j=1, j\neq i}^{n} (p_{ji}^{u})^{t} = 0 \ , \ \forall i, u = 2,...,n \ , \ i \neq u \ , \ \forall t = 1,...,m \tag{4.8k}$$

$$\sum_{j=1,\, j\neq u}^{n} (p_{uj}^{u})^{t} - \sum_{j=1,\, j\neq u}^{n} (p_{ju}^{u})^{t} = -z_{u}^{t} \ , \ \forall u = 2,...,n \ , \ \forall t = 1,...,m \qquad (4.8\ell)$$

$$x_{ij}^{t} = \{0,1\} \ , \ \forall i, j = 1,...,n \ , \ i \neq j \ , \ \forall t = 1,...,m \qquad (4.8\text{m})$$

$$z_{i}^{t} \geq 0 \ , \ \forall i = 2,...,n \ , \ \forall t = 1,...,m \qquad (4.8\text{n})$$

$$(p_{ij}^{u})^{t} \geq 0 \ , \ \forall i, j = 1,...,n \ , \ i \neq j \ , \ \forall u = 2,...,n \ , \ \forall t = 1,...,m \ . \qquad (4.8\text{o})$$

Here, $(p_{ij}^{u})^{t}$ represents the flow on arc (i, j) for the u^{th} commodity sent from node 1 to node u by some route $t = 1,...,m$. Moreover, we adopt the following additional valid restrictions, recognizing that $(p_{ij}^{u})^{t}$ represents the flow on arc (i, j) for the single unit of commodity u on some route $t = 1,...,m$, which flows from node 1 to node u.

$$(p_{j1}^{u})^{t} = 0 \ , \ \forall j, u = 2,...,n \ \ , \ \forall t = 1,...,m \ . \qquad (4.9\text{a})$$

$$(p_{uj}^{u})^{t} = 0 \ , \ \forall j, u = 2,...,n \ , \ j \neq u \ \ , \ \forall t = 1,...,m \ . \qquad (4.9\text{b})$$

$$(p_{1j}^{u})^{t} = z_{u}^{t} \cdot x_{1j}^{t} \ , \ \forall j, u = 2,...,n \ \ , \ \forall t = 1,...,m \ . \qquad (4.9\text{c})$$

$$(p_{ju}^{u})^{t} = z_{u}^{t} \cdot x_{ju}^{t} \ , \forall j, u = 2,...,n \ , \ j \neq u \ \ , \ \forall t = 1,...,m \ . \qquad (4.9\text{d})$$

Observe that by substituting (4.9) in the original formulation of mATSP-FL, we can make the following simplifications. First, by virtue of (4.9a,c), we have that (4.8j) equals to $z_{u}^{t} \cdot \left(\sum_{j=2}^{n} x_{1j}^{t} \right) = z_{u}^{t} \ , \ \forall u = 2,...,n \ \ , \ \forall t = 1,...,m$, which is implied by (4.8f). Hence, (4.8j) can be deleted. Similarly, using (4.9a-d) in (4.8ℓ), we have $-z_{u}^{t} \cdot \left(\sum_{j=1,\, j\neq u}^{n} x_{ju}^{t} \right) = -z_{u}^{t}$. It is implied by (4.8b), since $-z_{u}^{t} \cdot z_{u}^{t} = -z_{u}^{t}$ when z_{u}^{t} is binary, $\forall u = 2,...,n \ , \ \forall t = 1,...,m$. Therefore, (4.8ℓ) can also be omitted. Finally, by applying (4.9a,b,d), we can rewrite (4.8k) in the following strengthened form:

$$\left[\sum_{\substack{j=2 \\ j\notin\{i,u\}}}^{n} (p_{ij}^{u})^{t} + x_{iu}^{t} \right] - \left[\sum_{\substack{j=1 \\ j\notin\{i,u\}}}^{n} (p_{ji}^{u})^{t} \right] = 0 \ , \ \forall i, u = 2,...,n \ , \ i \neq u \ , \ \forall t = 1,...,m \ . \qquad (4.10)$$

Using these relationships in (4.8a-i), we can reformulate ATSP-FL as the following tightened version.

mATSP-FL2: Minimize $\sum_{i=1}^{n}\sum_{j=1,\,j\neq i}^{n} c_{ij}\cdot\left(\sum_{t=1}^{m} x_{ij}^{t}\right)$ (4.11a)

subject to

$$\sum_{i=1,i\neq j}^{n} x_{ij}^{t} = z_{j}^{t}\ ,\ \forall j=2,...,n\ ,\ \forall t=1,...,m \tag{4.11b}$$

$$\sum_{j=1,\,j\neq i}^{n} x_{ij}^{t} = z_{i}^{t}\ ,\ \forall i=2,...,n\ ,\ \forall t=1,...,m \tag{4.11c}$$

$$\sum_{t=1}^{m} z_{i}^{t} = 1\ ,\ \forall i=2,...,n \tag{4.11d}$$

$$\sum_{i=2}^{n} x_{i1}^{t} = 1\ ,\ \forall t=1,...,m \tag{4.11e}$$

$$\sum_{j=2}^{n} x_{1j}^{t} = 1\ ,\ \forall t=1,...,m \tag{4.11f}$$

$$x_{1j}^{t} \le z_{j}^{t}\ ,\ \forall j=2,...,n\ ,\ \forall t=1,...,m \tag{4.11g}$$

$$x_{i1}^{t} \le z_{i}^{t}\ ,\ \forall i=2,...,n\ ,\ \forall t=1,...,m \tag{4.11h}$$

$$(p_{ij}^{u})^{t} \le x_{ij}^{t}\ ,\ \forall i=1,...,n\ ,\ \forall j,u=2,...,n\ ,\ i\neq j\neq u\ ,\ \forall t=1,...,m \tag{4.11i}$$

$$\left[\sum_{\substack{j=2\\ j\notin\{i,u\}}}^{n}(p_{ij}^{u})^{t} + x_{iu}^{t}\right] - \left[\sum_{\substack{j=1\\ j\notin\{i,u\}}}^{n}(p_{ji}^{u})^{t}\right] = 0\ ,\ \forall i,u=2,...,n\ ,\ i\neq u\ ,\ \forall t=1,...,m \tag{4.11j}$$

$$x_{ij}^{t} = \{0,1\}\ ,\ \forall i,j=1,...,n\ ,\ i\neq j\ ,\ \forall t=1,...,m \tag{4.11k}$$

$$z_{i}^{t} \ge 0\ ,\ \forall i=2,...,n\ ,\ \forall t=1,...,m \tag{4.11ℓ}$$

$$(p_{ij}^{u})^{t} \ge 0\ ,\ \forall i,j=1,...,n\ ,\ i\neq j\ ,\ \forall u=2,...,n\ ,\ \forall t=1,...,m\ . \tag{4.11m}$$

We are, now, ready to show the following results.

Proposition 4.2. mATSP2 gives a tighter LP relaxation than mATSP-FL2.

Proof. Note that the constraints (4.11b-h,k,ℓ) of mATSP-FL2 are also present in mATSP2. Furthermore, let us rename the $\boldsymbol{p}$-variables in mATSP-FL2 as follows

$$(p_{ij}^{u})^{t} = (f_{iu}^{j})^{t},\ \forall i = 1,...,n,\ \forall j,u = 2,...,n,\ i \neq j \neq u\ ,\ \forall t = 1,...,m. \tag{4.12}$$

Then, (4.11i) and (4.11j) can, respectively, be re-written as follows:

$$(f_{iu}^{j})^{t} \leq x_{ij}^{t}\ ,\ \forall i = 1,...,n,\ \forall j,u = 2,...,n\ ,\ i \neq j \neq u\ ,\ \forall t = 1,...,m. \tag{4.13a}$$

$$\left[\sum_{j=2,\, j\notin\{i,u\}}^{n}(f_{iu}^{j})^{t} + x_{iu}^{t}\right] - \left[\sum_{j=1,\, j\notin\{i,u\}}^{n}(f_{ju}^{i})^{t}\right] = 0, \forall i,u = 2,...,n,\ i \neq u,\ \forall t = 1,...,m. \tag{4.13b}$$

Now, observe that (4.13a) is precisely (4.7a), and that, (4.13b) is obtained by considering (4.7b) written for y_{iu}^{t} on its right-hand-side and subtracting from this (4.7c) written for the same y_{iu}^{t} on its corresponding right-hand-side. Hence, mATSP-FL2 is a relaxed surrogate of mATSP2. This completes the proof. □

Proposition 4.3. mATSP2 is a valid formulation of the multiple salesmen asymmetric traveling salesman problem (mATSP).

Proof. The constraints defining mATSP2 are valid for mATSP since mATSP2 is a relaxation of mATSP1, which is a valid formulation due to the validity of mATSP0 and the RLT lifting process. Moreover, by Proposition 4.2, the constraints defining mATSP2 imply those of mATSP-FL2, which is a valid formulation of mATSP. This completes the proof. □

4.3.2.2 Extension of mATSP to the multiple-roller problem

In this section, we extend the mATSP formulations discussed above to the hot strip rolling scheduling problem for processing the slabs under the special type of precedence order (see Figure 4-2) simultaneously on multiple rollers. Suppose that there are n slabs that are to be rolled simultaneously using m turns. As explained earlier, the precedence relationship between any two slabs is considered if and only if those slabs are processed in the same turn. Let PC_j be the set of slabs that are required to precede slab j, $\forall j$. If $i \in PC_j$, then we denote this as $i \prec j$. We can accommodate the above special type of

precedence relationships in mATSP0, mATSP1, and mATSP2 by including the following two constraint sets.

$$y_{ij}^t \geq z_i^t + z_j^t - 1,\ \forall i \in PC_j,\ j = 2,...,n,\ i \neq j,\ \forall t = 1,...,m. \quad (4.14a)$$

$$x_{ji}^t = 0,\ \forall i \in PC_j,\ j = 2,...,n,\ \forall t = 1,...,m. \quad (4.14b)$$

The constraint set (4.14a) enforces slab i to be before slab j if and only if they are to be processed by the same roller t when $i \prec j$ is required. Constraint set (4.14b) implies that there can not exist any direct flow from slab j to slab i on any roller t when $i \prec j$ is required.

To accommodate the desired precedence relationships in mATSP-FL, we substitute $\sum_{j=1, j\notin\{i,u\}}^{n} (f_{ju}^i)^t$, or equivalently $\sum_{j=1, j\notin\{i,u\}}^{n} (p_{ji}^u)^t$ (see (4.12)), for y_{iu}^t from Proposition 4.2. The constraint set (4.14a) can, now, be rewritten as follows.

$$\sum_{u=1, u\notin\{i,j\}}^{n} (p_{ui}^j)^t \leq z_i + z_j - 1,\ \forall i \in PC_j,\ j = 2,...,n,\ i \neq j,\ \forall t = 1,...,m. \quad (4.15a)$$

Hence, we can accommodate the special type of precedence structure relevant for the processing of the slabs in mATSP-FL by including both (4.15a) and (4.14b).

For the mATSP–MTZ and mATSP–Bektas formulations, the precedence relationship between two slabs i and j can be captured explicitly by the following constraint sets.

$$u_j \geq u_i + 1,\ \forall i \in PC_j,\ j = 2,...,n. \quad (4.16a)$$

$$x_{ji} = 0,\ \forall i \in PC_j,\ j = 2,...,n. \quad (4.16b)$$

The real variables $u_i,\ \forall i = 2,...,n$, can be interpreted as the positions of the sequence in which the slabs are processed by a roller. The constraint set (4.16a) enforces the requirement that the position of slab i should be before slab j if $i \prec j$. However, it enforces this requirement even when slabs i and j ($i \prec j$) are not processed by the same roller. As a result, the precedence relationships among the slabs that are imposed by the above constraints are not the desired precedence constraints. Consequently, the inclusion of the desired precedence constraints for the processing of the slabs is not as straitght

forward in formulations mATSP-MTZ and mATSP-Bektas. On the other hand, we can easily incorporate the constraint sets (4.14a,b) in our new mATSP formulations, presented in Section 4.3.2.1, to capture the special type of precedence relationships required for the processing of the slabs in the hot strip rolling scheduling problem. We refer to the resulting formulations as rPCmATSP0, rPCmATSP1 and rPCmATSP2, respectively. Here, we use the tighter formulation mATSP2 to be a valid formulation for this scheduling problem.

rPCmATSP2: Minimize $\left\{ \sum_{i=2}^{n+1} \sum_{j=2, j\neq i}^{n+1} c_{ij} \cdot \left(\sum_{t=1}^{m} x_{ij}^{t} \right) \right.$: (4.3b)-(4.3h),(4.3j)-(4.3m), (4.3o,p), (4.7a)-(4.7c), and (4.14a,b) $\left. \right\}$.

4.3.2.3 Extension of the mATSP to the single-roller problem with slab selection

In this section, we extend the mATSP formulations to the hot strip rolling scheduling problem with slab selection under the special type of precedence order (see Figure 4-2). Note that, once a subset of slabs is selected for processing by a roller, the precedence order given in Figure 4-2 is to be maintained among these slabs irrespective of their relationships with the slabs that do not belong in this subset.

The prize collecting ATSP that is presented in the literature (see Balas [8]) to select a subset of cities, from a given set of cities, for a traveling salesman to visit, does not consider the precedence order in which the cites are to be visited. Here, we formulate this problem as a multiple-traveling salesman problem under the given restricted precedence constraints. This involves a slight modification of the formulations presented in the previous section.

Suppose that there are n slabs available to be rolled. We designate a dummy slab as $\{1\}$. We determine two sequences of slabs, connoting a two traveling salesmen problem, and assume that only one of the sequences is going to be used as real schedule for a turn. We define the precedence relationship as below. Let PC_j be the set of slabs that are required to precede slab j, $\forall j = 2,...,n+1$. If $i \in PC_j$, then we denote this as $i \prec j$. Referring to Figure 4-2, note that, once we select a subset of slabs to be processed in a

turn, we need to maintain the given precedence relationships only among the slabs that are selected for processing in that run, and not among the slabs belonging to different runs. This is thus a special case of the mATSP problem involving $m = 2$.

In order to enforce that a number of slabs are assigned to the first sequence, we consider a penalty cost, p_i, $\forall i = 2,...,n+1$, of putting a slab i in the second sequence, i.e., $c_{ii} = p_i$ for slab i, $\forall i = 2,...,n+1$. The objective is to minimize the total penalty incurred as a result of placing the slabs in both sequences. The penalty for placing the slabs in the first sequence is due to a change in the width, gauge and hardness from one slab to another, while the penalty pertaining to the second sequence is the result of having any slab assigned to it. Similar to rPCmATSP2, the modified two traveling salesmen problem with restricted precedence constraints is as follows.

rPC2ATSP2: Minimize $\left\{ \sum_{i=2}^{n+1} \sum_{j=2, j\neq i}^{n+1} c_{ij} \cdot x_{ij}^1 + \sum_{i=2}^{n+1} c_{ii} \cdot z_i^2 : \text{(4.3b)-(4.3h),(4.3j)-(4.3m), (4.3o,p), (4.7a)-(4.7c), and (4.14a,b)} \right\}$.

4.3.2.4 Extension of the mATSP to the multiple-roller problem with slab selection

In this section, we extend the mATSP formulations to the hot strip rolling scheduling problem with slab selection under the special type of precedence order (see Figure 4-2) simultaneously on multiple rollers. Suppose that there are n slabs available to be rolled simultaneously on m turns. We determine $m+1$ sequences of slabs, or solve a $m+1$-traveling salesman problem, where the first m of the sequences are going to be used as real schedules for m turns. Also, we consider a penalty cost, p_i, $\forall i = 2,...,n+1$, of putting a slab i on the $(m+1)^{\text{th}}$ roller, i.e., $c_{ii} = p_i$, $\forall i = 2,...,n+1$. The objective is to minimize the total penalty cost, where the penalty for the first m sequences constitutes the penalty due to a change in the width, gauge and hardness from one slab to another, and the penalty for the $(m+1)^{\text{th}}$ sequence is the result of having any slab assigned to it. We can modify rPCmATSP2 to include the $m+1$ traveling salesmen under restricted precedence constraints as follows.

rPC(m+1)ATSP2: Minimize $\left\{ \sum_{i=2}^{n+1} \sum_{j=2, j\neq i}^{n+1} c_{ij} \cdot \left(\sum_{t=1}^{m} x_{ij}^{t} \right) + \sum_{i=2}^{n+1} c_{ii} \cdot z_i^{m+1} : \text{(4.3b)-(4.3h),(4.3j)-(4.3m), (4.3o,p), (4.7a)-(4.7c), and (4.14a,b)} \right\}$.

4.4 Computational Results

Our computational analysis includes three sections. In Section 4.4.1, we consider the multiple-asymmetric traveling salesman problem (mATSP) and compare the LP relaxation bounds of different mATSP formulations. We use the standard test problems for the asymmetric traveling salesman problem (ATSP) from the TSP library and extend those to the determination of m tours, each commencing from the base city. In Section 4.4.2, we consider the mATSP involving the restricted precedence structure (rPCmATSP) and compare the LP relaxation bounds of different rPCmATSP formulations. For this experimentation, we take the standard test problems for the precedence-constrained ATSP (PCATSP) from the TSP library as well as two sets of randomly generated test problems and extend those to include m tours commencing from the base city. Then, in Section 4.4.3, we use randomly generated test problems with adjustable density ratio of precedence network for the study of the effect of various density ratios in solving the hot strip rolling scheduling problems.

4.4.1 Computational comparison of various mATSP formulations

In this section, we present computational results to compare the LP relaxation bounds of our new formulations mATSP0, mATSP1, and mATSP2 with three other formulations, namely, mATSP–MTZ, mATSP–Bektas and mATSP–FL. For the compatibility of results, we assume the values of K and L to be 1 and n, respectively, for the mATSP–Bektas model (see Section 2.2).

For this experimentation, we consider four standard test problems from the TSP library (TSPLIB), namely br17, ftv33, ftv35 and ftv38. We add another parameter, m, the total number of salesmen available, to these data sets. All runs were made on a Dell Workstation PWS 650 with double 2.5GHz CPU Xeon processors and 1.5GB RAM, running the Windows XP operating system. We used AMPL version 8.1 and CPLEX

MIP Solver version 9.0. Note that every salesman has to visit at least one city starting from the base city.

Table 4-1 Results pertaining to the LP relaxations of the various mATSP formulations

Problem	m		mATSP-MTZ	mATSP-Bektas	mATSP-FL	mATSP0	mATSP1	mATSP2
br17	2	z_{LP}	8.13	22.00	**39.00**	6.00	**39.00**	**39.00**
		%Gap	79.2	43.6	0	84.6	0	0
		CPU	0.03	0.01	10.05	0.34	65.15	16.71
	3	z_{LP}	14.29	28.00	**42.00**	12.00	**42.00**	**42.00**
		%Gap	66.0	33.3	0	71.4	0	0
		CPU	0.01	0.02	19.33	0.53	137.32	35.07
	4	z_{LP}	20.46	34.00	**47.00**	18.00	**47.00**	**47.00**
		%Gap	56.5	27.7	0	61.7	0	0
		CPU	0.01	0.02	36.53	0.52	89.80	39.67
ftv33	2	z_{LP}	1211.13	1234.57	**1302.00**	1209.00	**1302.00**	**1302.00**
		%Gap	7.0	5.2	0	7.1	0	0
		CPU	0.02	0.01	581.11	2.31	18,684.00	850.86
	3	z_{LP}	1238.03	1260.54	**1328.00**	1236.00	**1328.00**	**1328.00**
		%Gap	6.8	5.1	0	6.9	0	0
		CPU	0.03	0.01	2,001.25	13.08	69,659.70	7,834.23
	4	z_{LP}	1266.10	1302.93	**1367.00**	1263.00	**1367.00**	**1367.00**
		%Gap	7.4	4.7	0	7.6	0	0
		CPU	0.02	0.02	3,200.84	12.09	211,467.00	28,911.60
ftv35	2	z_{LP}	1395.47	1425.79	**1466.50**	1393.00	**1466.50**	**1466.50**
		%Gap	6.3	4.2	1.5	6.4	1.5	1.5
		CPU	0.02	0.02	543.69	2.41	86,599.10	1,185.70
	3	z_{LP}	1422.03	1467.41	**1495.50**	1419.00	**1495.50**	**1495.50**
		%Gap	5.9	2.9	1.0	6.1	1.0	1.0
		CPU	0.03	0.02	2,471.06	710.95	290,547.00	14,609.30
	4	z_{LP}	1464.19	1511.25	**1539.50**	1461.00	**1539.50**	**1539.50**
		%Gap	5.6	2.6	0.7	5.8	0.7	0.7
		CPU	0.02	0.02	6,094.46	494.31	247,110.00	56,724.40
ftv38	2	z_{LP}	1431.22	1458.35	**1492.00**	1429.00	**1492.00**	**1492.00**
		%Gap	4.9	3.1	0.9	5.0	0.9	0.9
		CPU	0.02	0.02	535.97	7.22	61,227.30	2,310.52
	3	z_{LP}	1456.88	1471.61	**1504.50**	1455.00	**1504.50**	**1504.50**
		%Gap	4.2	3.2	1.1	4.3	1.1	1.1
		CPU	0.02	0.02	3,646.61	238.64	134,578.00	12,614.10
	4	z_{LP}	1485.13	1515.66	**1540.50**	1483.00	**1540.50**	**1540.50**
		%Gap	3.9	2.0	0.4	4.1	0.4	0.4
		CPU	0.02	0.02	8,704.25	496.23	269,141.00	34,488.80

The results, shown in Table 4-1, include the lower bound value (z_{LP}) obtained by solving the LP relaxation, the percentage gap between the LP-based lower bound and the optimal integer solution (%Gap), and the cpu time (in seconds) required to obtain the optimal LP solution for $m = 2$, 3, and 4. Note that the number in bold indicates the highest z_{LP} among all models.

From Table 4-1, the tightest lower bounds (z_{LP}) are obtained via the LP relaxations of mATSP–FL, mATSP1, and mATSP2. Recall that, in Proposition 4.2, we have shown mATSP2 to be at least as tight as a LP relaxation as mATSP-FL2, which is equivalent to mATSP-FL. All lower bound values generated by these formulations turn out to be the same for this set of problems. However, their lower bounds are significantly better than those obtained by mATSP-MTZ and mATSP-Bektas, thus exhibiting their tightness.

Further, we exploit two mixed-integer programming algorithms to solve mATSP2: the branch and bound algorithm implemented by CPLEX (under default option; in the sequel, we refer to it as the CPLEX algorithm, or simply CPLEX) and a Benders´ decomposition method [13]. The implementation of Benders´ decomposition for solving mATSP2 is as follows.

Benders´ decomposition method for solving mATSP2

In accordance with the Benders´ decomposition method, the primal mATSP2 model is decomposed into two parts: a reduced master problem (RMP) and an associated sub-problem (SP). Due to the special nature of our formulation, the sub-problem (SP) can be split into m separate sub-problems, corresponding to m salesmen. The procedure is as follows.

Initialization:

Construct the first reduced master problem (RMP_0) consisting of binary variables ***x*** and ***z*** with the objective function (4.3a) and the constraint sets (4.3b)-(4.3h). We solve RMP_0 to optimality and obtain the first integer solution $(\bar{\mathbf{x}}_1, \bar{\mathbf{z}}_1)$. Set $k = 1$.

Main step:

At iteration k, we construct m sub-problems, $SP_t(\bar{\mathbf{x}}_k, \bar{\mathbf{z}}_k)$, $t = 1,...,m$, for m traveling salesmen corresponding to the associated continuous variables $\boldsymbol{y}$ and $\boldsymbol{f}$ with a dummy objective function and the following constraint sets: (4.3j)-(4.3m) and (4.7a)-(4.7c). These constraints are augmented by the fixed variable value $(\bar{\mathbf{x}}_k, \bar{\mathbf{z}}_k)$. If all sub-problems can be solved to optimality, stop and the current integer solution $(\bar{\mathbf{x}}_k, \bar{\mathbf{z}}_k)$ is optimal to the original problem; otherwise, m Benders´ cuts are added to RMP_k. Increase k to $k+1$. Solve the current reduced master problem (RMP_k) and obtain an optimal integer solution $(\bar{\mathbf{x}}_k, \bar{\mathbf{z}}_k)$. Repeat the main step.

Note that, since each sub-problem $SP_t(\bar{\mathbf{x}}_k, \bar{\mathbf{z}}_k)$, $t = 1,...,m$ is separate from the other, we can remove the salesman index t from both $\boldsymbol{y}$ and $\boldsymbol{f}$ variables.

Table 4-2 Results pertaining to the IP solution of mATSP2 by CPLEX and Benders´ decomposition

Problem	m	CPLEX			Benders´ Decomposition Method		
		Best Integer	CPU time (IP)	Nodes	Best Integer	CPU time (IP)	Cuts
br17	2	39	98	0	39	51	36
	3	42	178	0	42	275	53
	4	47	137	0	47	17,710	76
ftv33	2	1302	843	0	1302	1,855	21
	3	1328	2,900	0	1328	2,626	29
	4	1367	6,086	0	1367	1,931	27
ftv35	2	1489	25,265	167	1489	26,915	55
	3	1541	35,000*	1	1511	16,832	43
	4	-**	35,000*	0	1551	15,237	64
ftv38	2	1551	35,000*	24	1505	28,230	37
	3	1567	35,000*	0	1521	25,355	46
	4	-**	35,000*	0	1546	10,622	32

- * Exceed time limit (35,000 seconds).
- ** No integer solution is obtained within time limit.

We solved the standard test problems from the TSP library using both CPLEX and the Benders´ decomposition method described above. The results are shown in Table 4-2. For CPLEX, we present the best integer solution (Best Integer), the total cpu time (in seconds) required to obtain the optimal IP solution, and the number of nodes (Nodes) generated by CPLEX; and, for the Benders´ decomposition method, we present the best

integer solution (Best Integer), the total cpu time (in seconds) required to obtain the optimal IP solution, and the number of Benders´ iterations (Cuts).

For larger size problems, such as ftv35 and ftv38, Benders´ decomposition method generally requires less cpu time. For the test problems that we used, we are able to obtain optimal solutions using Benders´ decomposition method while that is not the case for CPLEX. For CPLEX, we are either not able to obtain a solution in some cases, or we obtain an inferior solution within a pre-specified cpu time.

4.4.2 Computational comparison of various rPCmATSP formulations

In this section, we present computational results that compare the performances of the following formulations for the restricted precedence constrained mATSP (rPCmATSP): rPCmATSP-FL, rPCmATSP1, and rPCmATSP2.

We used two sets of data for our experimentation. The first set consists of four problems from the TSP library (TSPLIB), namely, esc07, esc12, br17.10, and esc25. We add another parameter, m, the total number of salesmen available to these data sets. The results are shown in Tables 4-3 that include the lower bound value (z_{LP}) obtained by solving the LP relaxation, and the cpu time (in seconds) required to obtain the optimal LP solution, for $m = 2$, 3, and 4.

Table 4-3 Results pertaining to the LP relaxations of the various rPCmATSP formulations

Problem	m		rPCmATSP-FL	rPCmATSP1	rPCmATSP2	Problem	m		rPCmATSP-FL	rPCmATSP1	rPCmATSP2
esc07	2	z_{LP}	650	650	650	br17.10	2	z_{LP}	39	39	39
		CPU	0.02	0.06	0.03			CPU	13.05	62.34	17.47
	3	z_{LP}	325	325	325		3	z_{LP}	42	42	42
		CPU	0.02	0.03	0.02			CPU	27.92	108.20	25.31
	4	z_{LP}	100	100	100		4	z_{LP}	47	47	47
		CPU	0.03	0.05	0.03			CPU	40.84	121.92	30.76
esc12	2	z_{LP}	1209	1209	1209	esc25	2	z_{LP}	1141	1141	1141
		CPU	0.58	2.47	0.95			CPU	74.77	1,733.74	113.06
	3	z_{LP}	1015.5	1015.5	1015.5		3	z_{LP}	943	943	943
		CPU	1.64	4.39	1.41			CPU	89.59	10,246.10	178.13
	4	z_{LP}	822	822	822		4	z_{LP}	795	795	795
		CPU	2.08	2.33	1.50			CPU	141.00	30,352.50	251.58

The second data consists of two sets of randomly generated problems having 19 and 24 cities, respectively, with each set consisting of three test problems containing different number of precedence relations in order to capture different density ratios in a precedence network. The density ratio of a precedence network is defined here as a ratio of the number of direct precedence relations divided by [(n-1)*(n-2)/2], which is the maximum total number of direct precedence relations, not including the base city, in a network of n cities. Specifications for these randomly generated problem data sets are displayed in Table 4-4. The random problem generator used is adapted from a Microsoft foundation class (MFC) application named 'ATSP_Prec'. The distances between cities are integer numbers ranging from 0 to 100.

Table 4-4 Randomly generated problem data sets

Problem	Number of Precedence Relations	Density (%)	Problem	Number of Precedence Relations	Density (%)
19_1	6	3.9	24_1	10	3.9
19_2	22	14.4	24_2	35	13.8
19_3	52	34.0	24_3	83	32.8

Table 4-5 presents the results on the lower bound value (z_{LP}) obtained by solving the LP relaxations of the formulations, and the cpu time (in seconds) required to obtain the optimal LP solution.

Table 4-5 Results pertaining to the LP relaxations of the various rPCmATSP formulations for randomly generated problems

Problem	m		rPCmATSP-FL	rPCmATSP1	rPCmATSP2	Problem	m		rPCmATSP-FL	rPCmATSP1	rPCmATSP2
19_1	2	z_{LP}	170	170	170	24_1	2	z_{LP}	163	163	163
		CPU	2.61	14.13	3.52			CPU	22.49	742.93	48.63
	3	z_{LP}	186	186	186		3	z_{LP}	178	178	178
		CPU	4.52	418.69	5.69			CPU	89.05	3,616.79	193.42
	4	z_{LP}	210	210	210		4	z_{LP}	208.5	208.5	208.5
		CPU	8.80	1,714.96	5.92			CPU	140.30	16,979.70	458.01
19_2	2	z_{LP}	182	182	182	24_2	2	z_{LP}	150	150	150
		CPU	10.30	60.20	13.94			CPU	34.55	721.07	45.63
	3	z_{LP}	207.25	207.25	207.25		3	z_{LP}	172	172	172
		CPU	25.31	100.44	19.59			CPU	93.41	6,630.89	154.62
	4	z_{LP}	237	237	237		4	z_{LP}	201	201	201
		CPU	26.14	803.48	14.78			CPU	54.64	4,915.86	100.89
19_3	2	z_{LP}	226	226	226	24_3	2	z_{LP}	218.5	218.5	218.5
		CPU	2.14	11.44	3.13			CPU	27.58	494.36	40.89
	3	z_{LP}	254	254	254		3	z_{LP}	240	240	240
		CPU	17.66	68.87	14.91			CPU	33.63	3,351.32	51.22
	4	z_{LP}	293	293	293		4	z_{LP}	270	270	270
		CPU	35.27	652.20	27.05			CPU	47.52	13,363.80	46.86

Referring to Table 4-3 and Table 4-5, note that all three formulations provide the same lower bound values. However, it is apparent that the cpu time required by rPCmATSP1 is dominated by those required by rPCmATSP-FL and rPCmATSP2.

As we did in Section 4.4.1, we used two mixed-integer programming methods to solve rPCmATSP2 to optimality: CPLEX and Benders´ decomposition. Furthermore, we investigate the implementation of a modified version of the Benders´ decomposition method, referred to as Modified Benders´, which was developed to further improve its computational performance. The focus of this variation is to decrease the number of reduced master problems by generating as many Benders´ cuts as possible at the first node.

Modified Benders´ method for solving rPCmATSP2

In the initialization step, we construct the first reduced master problem (RMP_0) consisting of binary variables ***x*** and ***z*** for the objective function (4.3a) and the constraint sets (4.3b)-(4.3h) and (4.14b).

Initialization:

Solve the LP relaxation of RMP_0 and obtain the optimal LP solution $(\bar{\mathbf{x}}_1, \bar{\mathbf{z}}_1)$.

Phase I:

At iteration k, we construct m sub-problems, $SP_t(\bar{\mathbf{x}}_k, \bar{\mathbf{z}}_k)$, $t = 1,...,m$, for m traveling salesmen by using the associated continuous variables ***y*** and ***f*** with a dummy objective function and the following constraint sets: (4.3j)-(4.3m) and (4.7a)-(4.7c). These constraints are augmented by the fixed variable value $(\bar{\mathbf{x}}_k, \bar{\mathbf{z}}_k)$. If all sub-problems can be solved to optimality, go to Phase II; otherwise, m Benders´ cuts are added to RMP_k. Increase k to $k+1$. Solve the LP relaxation of current reduced master problem (RMP_k) and obtain an optimal LP solution $(\bar{\mathbf{x}}_k, \bar{\mathbf{z}}_k)$. Repeat Phase I.

Phase II:

Solve RMP_k to optimality and obtain an integer solution $(\bar{\mathbf{x}}_k, \bar{\mathbf{z}}_k)$. Then, construct m sub-problems, $SP_t(\bar{\mathbf{x}}_k, \bar{\mathbf{z}}_k)$, $t = 1,...,m$, for m traveling salesmen by using the associated continuous variables ***y*** and ***f*** with a dummy objective function and the following constraint sets: (4.3j)-(4.3m), (4.7a)-(4.7c), and (4.14a). These constraints are augmented by the fixed variable value $(\bar{\mathbf{x}}_k, \bar{\mathbf{z}}_k)$. If all sub-problems can be solved to optimality, stop; and, the current integer solution $(\bar{\mathbf{x}}_k, \bar{\mathbf{z}}_k)$ is optimal; otherwise, m Benders´ cuts are added to RMP_k. Increase k to $k+1$. Solve current reduced master problem (RMP_k) and obtain an optimal IP solution $(\bar{\mathbf{x}}_k, \bar{\mathbf{z}}_k)$. Repeat Phase II.

The results of our experimentation are shown in Table 4-6 and Table 4-7. Table 4-6 contains results for the problems from the TSP library while Table 4-7 contains results for the randomly generated problems. For CPLEX, we present the total cpu time (in seconds) required to obtain the best IP solution, and the required number of branching nodes (Nodes). For Benders´ decomposition and Modified Benders´ methods, we present the total cpu time (in seconds) required to obtain the best IP solution, and the number of Benders´ iterations (Cuts) required.

Table 4-6 Results pertaining to the IP solution of rPCmATSP2 by CPLEX, Benders´ decomposition and Modified Benders´ methods (for problems from the TSP library)

Problem	*m*	CPLEX			Benders´ Decomposition			Modified Benders´		
		Best Integer	CPU time (IP)	Nodes	Best Integer	CPU time (IP)	Cuts	Best Integer	CPU time (IP)	Cuts
esc07	2	900	0.5	6	900	0.2	7	900	1.4	10
	3	450	0.6	5	450	0.2	6	450	1.3	7
	4	200	0.1	0	200	0.1	2	200	2.3	10
esc12	2	1270	17	12	1270	4	24	1270	5	22
	3	1021	16	0	1021	1	7	1021	5	23
	4	822	1	0	822	1	4	822	6	21
br17.10	2	46	11,303	1,066	46	5,641	391	46	8,870	335
	3	46	5,487	363	46	14,969	273	46	791	160
	4	49	10,352	442	49	4,211	131	49	993	132
esc25	2	1163	2,788	71	1163	42	10	1163	50	25
	3	980	1,497	18	980	39	12	980	26	15
	4	832	1,681	25	832	59	14	832	62	18

From Table 4-6, we can conclude that both Benders´ decomposition and Modified Benders´ methods are faster than CPLEX. Between Benders´ decomposition and Modified Benders´, the difference is subtle. For the test problem br17.10 with the number of traveling salesman 3 and 4, the Modified Benders´ method requires significantly less cpu time than that required by Benders´ decomposition method; while for several other instances, the Benders´ decomposition method requires less cpu time.

Table 4-7 Results pertaining to the IP solution of rPCmATSP2 by CPLEX, Benders´ decomposition and Modified Benders´ methods (for randomly generated problems)

Problem	*m*	CPLEX			Benders´ Decomposition			Modified Benders´		
		Best Integer	CPU time (IP)	Nodes	Best Integer	CPU time (IP)	Cuts	Best Integer	CPU time (IP)	Cuts
19_1	2	179	77	7	179	3	6	179	3	6
	3	186	4	0	186	1	1	186	3	8
	4	210	4	0	210	1	2	210	2	3
19_2	2	245	14,368	3,834	245	614	127	245	799	104
	3	248	14,905	3,536	248	1,269	103	248	2,269	132
	4	252	4,262	1,167	252	147	33	252	164	43
19_3	2	265	2,096	610	265	83	52	265	101	55
	3	277	1,737	329	277	134	45	277	114	50
	4	311	4,618	1,156	311	244	56	311	377	64
24_1	2	169	2,612	20	169	60	9	169	19	12
	3	181	2,017	4	181	25	8	181	32	18
	4	212	2,835	186	212	54	12	212	41	26
24_2	2	178	37,546	805	178	425	72	178	379	67
	3	195	86,400*	2,373	195	727	67	195	583	64
	4	223	42,052	1,842	223	482	44	223	603	53
24_3	2	424	86,400*	2,469	342	5,786	192	342	6,395	194
	3	359	135,000*	1,790	341	134,366	152	341	89,812	128
	4	375	110,000*	3,284	361	97,023	105	361	102,382	127

- * Exceed time limit.

Table 4-7 shows similar types of results. However, for the 24-3 problem, within the specified cpu time, both Benders´ decomposition and Modified Benders´ methods obtain optimal solutions while CPLEX does not. Thus, the performance of CPLEX appears to deteriorate faster.

4.4.3 Computational comparison of the Hot Strip Rolling Scheduling Problem

First, we discuss generation of test problems that capture the essence of a hot strip rolling scheduling problem. A key information in this regard is the special type of precedence order in which the slabs are to be processed. We discuss this in detail. This is followed by our investigation of the tightness of the models presented in Section 4.3.2 for this problem.

Given a set of slabs to be scheduled, we can sort these slabs in the non-increasing order of their widths as, ideally, required for their processing. This is depicted in Figure 4-5. We can use a matrix to represent the precedence relationships among the slabs. The value of element (i, j) of this matrix is 1 if slab i precedes slab j. If a strict precedence restriction is applied to these slabs, then all entries in the upper triangle of this matrix will be 1s. Consequently, the density ratio of the underlying precedence network is 1 (or 100%).

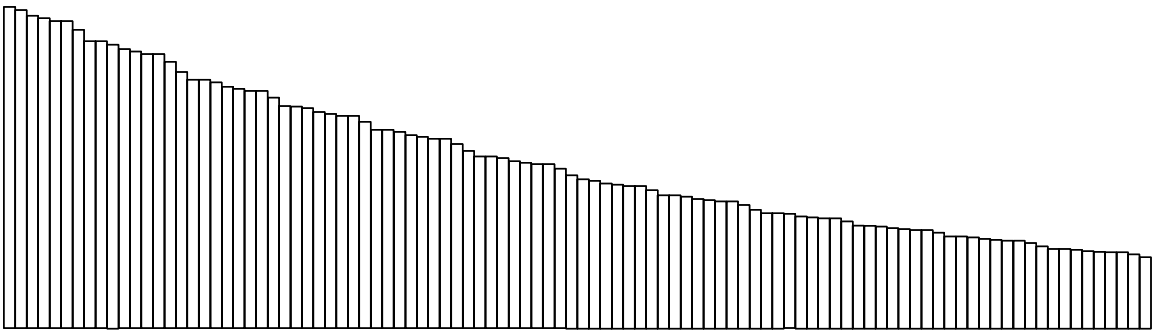

Figure 4-5 A non-increasing order in width of available slabs

However, in reality, the width of the slabs is not the only factor used to claim the feasibility of a hot rolling strip schedule. Besides, several slabs can have either the same width or the differences among their widths can be very small. Therefore, a strict adherence to the non-increasing widths of the slabs need not be maintained for their processing. To that end, we define the concept of moving tolerance window for slab widths. Let p be a moving tolerance ratio between 0 and 1. If there are, in all, n slabs available to be rolled, then the tolerance window for every one of the slabs, which are arranged in the non-increasing order of their widths, is $\lfloor p \cdot (n-1) \rfloor$. For example, if $p = 0.2$, the tolerance window is $\lfloor 0.2 \cdot (n-1) \rfloor$. This is shown in Figure 4-6 by non-shaded slabs for the first of the slabs that are arranged in the non-increasing order of their

widths. Note that, as a result of this tolerance window, the non-shaded slabs can be processed either before or after the first (widest) slab. In other words, the precedence relations among these slabs are relaxed. The same concept of tolerance window is applied to every one of the slabs, which are arranged in the non-increasing order of their widths.

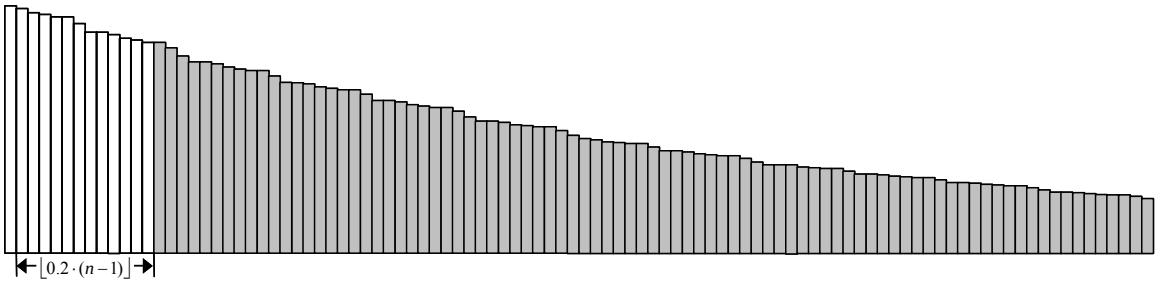

Figure 4-6 The succeeding slabs of the first (widest) slabs when the moving tolerance ratio $p = 0.2$

We can show the following result.

Proposition 4.4. The network density of the precedence relationships matrix for the slabs with moving tolerance ratio p is bounded by $\dfrac{((1-p)\cdot n + p - 1)\cdot((1-p)\cdot n + p)}{n\cdot(n-1)}$, for $0 < p \le 1$ and $n \ge 1$.

Proof. Let $k = \lfloor p \cdot (n-1) \rfloor$ be the number of slabs in the tolerance window with moving tolerance ratio p. Then, the total number of precedence relationships between the slabs can be calculated as follows.

$$\sum_{i=1}^{n}\left(|n_i| - k\right)^+ \ge \sum_{i=1}^{n}\left(n - i - p\cdot(n-1)\right)^+ = \sum_{i=1}^{\min\{(1-p)\cdot n + p, n\}}\left(n - i - p\cdot(n-1)\right)$$
$$= \frac{((1-p)\cdot n + p)((1-p)\cdot n + p - 1)}{2}, \text{ for any } 0 < p \le 1 \text{ and } n \ge 1 . \qquad (4.17)$$

Dividing (4.17) by $n\cdot(n-1)/2$, which is the maximum number of precedence relations among n slabs, we have the network density of the precedence matrix for the slabs, with moving tolerance ratio p, bounded by $\dfrac{((1-p)\cdot n + p)((1-p)\cdot n + p - 1)}{n(n-1)}$. □

Remark 4.1. The network density with moving tolerance ratio p converges to $(1-p)^2$ for a larger value of n.

From Remark 4.1, we can estimate the network density from the moving tolerance window p. For example, if the moving tolerance window p is 0.05, the estimated density ratio is at least 90.25%.

In the following computational experimentation, we use a set of five randomly generated problems including 30 slabs. Furthermore, we apply 8 different network densities on these test problems to study the impact of network density on computational performance. The network densities used are 90.25% (p=0.05), 81% (p=0.1), 72.25% (p=0.15), 64% (p=0.2), 56.25% (p=0.25), 20.25% (p=0.55), 9% (p=0.70), and 1% (p=0.9). The results presented in Table 4-8, Table 4-9 and Table 4-10 include the cpu time (limited to 18,000 cpu seconds) required to obtain the IP solution by using CPLEX, the total cpu time (in seconds) and the number of iterations (Cuts) required to obtain the IP solution by using Benders´ decomposition and Modified Benders´ methods, for two-roller, three-roller and four-roller problems, respectively.

Table 4-8 Results pertaining to IP solution of rPCmATSP2 by the Benders´ decomposition and Modified Benders´ methods (two rollers)

Problem	Density (%)	CPLEX	Benders´ Decomposition		Modified Benders´		Density (%)	CPLEX	Benders´ Decomposition		Modified Benders´	
		CPU(s)	CPU (s)	Cuts	CPU (s)	Cuts		CPU(s)	CPU (s)	Cuts	CPU (s)	Cuts
30_1	90.25	33	1	0	3	0	1.00	14,907	750	17	318	18
30_2		27	1	0	6	0		586	238	9	210	12
30_3		36	2	0	5	0		8,758	186	6	151	7
30_4		29	1	0	5	0		18,000*	334	11	423	18
30_5		26	2	0	4	0		18,000*	474	13	600	17
30_1	81.00	789	31	10	25	10	9.00	18,000*	469	21	496	33
30_2		569	41	16	33	15		18,000*	2,879	66	5,841	81
30_3		1,325	48	17	41	18		18,000*	1,212	43	1,262	56
30_4		12,271	51	19	68	27		18,000*	6,455	126	6,477	134
30_5		14,556	55	22	70	26		4,250	425	15	222	16
30_1	72.25	290	37	13	150	36	20.25	18,000*	11,232	98	0,827	113
30_2		18,000*	520	56	284	51		18,000*	1,647	50	2,048	56
30_3		18,000*	1,136	55	377	56		18,000*	1,905	58	1,424	49
30_4		18,000*	1,975	108	1,888	93		16,911	800	33	640	39
30_5		18,000*	145	28	211	39		18,000*	1,788	63	1,807	73
30_1	64.00	18,000*	443	67	414	68	56.25	18,000*	562	60	543	61
30_2		18,000*	817	54	1,110	78		18,000*	119	19	100	28
30_3		18,000*	308	45	336	48		18,000*	3,076	109	2,317	103
30_4		18,000*	205	43	171	40		18,000*	614	61	582	67
30_5		18,000*	963	89	1,065	93		18,000*	646	58	646	57

- * Exceed time limit (18,000 seconds).

Table 4-9 Results pertaining to IP solution of rPCmATSP2 by the Benders´ decomposition and Modified Benders´ methods (three rollers)

Problem	Density (%)	CPLEX	Benders´ Decomposition		Modified Benders´		Density (%)	CPLEX	Benders´ Decomposition		Modified Benders´	
		CPU(s)	CPU (s)	Cuts	CPU (s)	Cuts		CPU(s)	CPU (s)	Cuts	CPU (s)	Cuts
30_1	90.25	33	2	0	4	0	1.00	970	202	9	52	13
30_2		35	1	0	4	0		292	23	1	29	2
30_3		199	2	0	6	0		5,488	131	10	37	5
30_4		53	2	0	4	0		9,425	539	8	296	25
30_5		64	1	0	6	0		143	34	2	12	3
30_1	81.00	4,122	46	11	36	10	9.00	18,000*	352	20	301	40
30_2		927	29	9	27	13		18,000*	9,274	77	37,175	121
30_3		18,000*	76	14	45	10		18,000*	469	29	674	43
30_4		788	25	8	15	6		18,000*	4,050	110	20,592	122
30_5		3,797	54	13	50	20		2,562	413	26	221	21
30_1	72.25	327	10	2	18	8	20.25	18,000*	76,553	130	145,013	137
30_2		18,000*	308	34	260	35		18,000*	0,676	66	7,667	66
30_3		18,000*	1,245	42	839	54		18,000*	242	20	220	19
30_4		18,000*	1,301	58	2,406	77		18,000*	1,189	37	927	41
30_5		18,000*	573	25	928	22		18,000*	2,561	76	1,659	61
30_1	64.00	18,000*	136	24	368	58	56.25	18,000*	3,413	57	2,067	60
30_2		18,000*	1,796	49	1,534	58		18,000*	497	35	201	29
30_3		18,000*	358	33	295	48		18,000*	9,372	67	12,783	77
30_4		18,000*	138	16	81	24		18,000*	159	17	377	33
30_5		18,000*	3,704	34	2,007	39		18,000*	1,733	42	1,322	41

- * Exceed time limit (18,000 seconds).

Table 4-10 Results pertaining to IP solution of rPCmATSP2 by the Benders´ decomposition and Modified Benders´ methods (four rollers)

Problem	Density (%)	CPLEX	Benders´ Decomposition		Modified Benders´		Density (%)	CPLEX	Benders´ Decomposition		Modified Benders´	
		CPU(s)	CPU (s)	Cuts	CPU (s)	Cuts		CPU(s)	CPU (s)	Cuts	CPU (s)	Cuts
30_1	90.25	39	2	0	4	0	1.00	6,884	134	11	59	18
30_2		62	2	0	5	0		447	37	2	16	4
30_3		423	2	0	5	0		18,000*	157	13	100	26
30_4		67	2	0	5	0		18,000*	510	21	530	56
30_5		119	2	0	5	0		441	45	4	32	9
30_1	81.00	18,000*	120	13	65	15	9.00	18,000*	2,483	47	1,608	53
30_2		2,382	72	13	70	17		18,000*	12,971	57	21,825	81
30_3		18,000*	137	13	67	12		18,000*	577	39	363	28
30_4		1,076	6	1	12	3		18,000*	1,528	51	1,301	59
30_5		18,000*	169	18	100	26		14,999	466	26	332	25
30_1	72.25	2,468	16	3	23	8	20.25	18,000*	16,808	69	15,230	69
30_2		18,000*	906	33	605	36		18,000*	10,426	53	13,502	68
30_3		18,000*	815	34	460	39		18,000*	4,721	44	3,289	52
30_4		18,000*	1,729	58	1,265	54		18,000*	174	9	570	24
30_5		18,000*	5,590	22	73,163	33		18,000*	330	18	226	16
30_1	64.00	18,000*	666	23	249	26	56.25	18,000*	8,202	55	9,750	64
30_2		18,000*	1,694	39	1,428	50		18,000*	473	28	476	32
30_3		18,000*	249	24	130	25		18,000*	137,173	86	115,910	77
30_4		18,000*	552	24	668	27		18,000*	2,599	51	3,325	68
30_5		18,000*	6,213	32	5,069	26		18,000*	2,312	33	1,709	29

- * Exceed time limit (18,000 seconds).

From Table 4-8, Table 4-9 and Table 4-10, we can conclude that both Benders´ decomposition and Modified Benders´ methods consistently perform better than CPLEX. Both of these methods can solve all except two test problems within 5 cpu hours (18,000 cpu seconds) while CPLEX can not solve 75 of the 120 test problems within this cpu time. Moreover, the results show that when the network densities are 90.25% (p=0.05), 81% (p=0.1), and 1% (p=0.9), all five test problems can be solved within 1,000 seconds by Benders´ decomposition and Modified Benders´ methods. When the network density is 56.25% (p=0.25) or 20.25% (p=0.55), both of these methods take a little longer to solve the associated test problems, while CPLEX cannot solve most of these problems within the specified cpu time.

Chapter 5

A Compact Formulation for the Job Shop Scheduling Problem

This chapter presents a compact mathematical formulation for the job shop scheduling problem. We begin in Section 5.1 by reviewing several mathematical formulations for the job shop scheduling problem. In Section 5.2, we present a new formulation for the job shop scheduling problem. We demonstrate its implementation on two test problems given in Muth and Thompson [53], in Section 5.3.

5.1 Literature on the Mathematical Programming Formulations for the Job Shop Scheduling Problem

In a job shop, a given set of jobs is processed on a set of available machines. Each job has its own route through the machines and a job may visit a machine more than once. The problem is to determine a schedule for processing the jobs on the machines so that the value of a given objective function is optimized. We consider the objective of minimizing the total time (also termed makespan) required to process all the jobs.

The job shop scheduling problem can be formulated as a time indexed model or based on a representation that relies on a conjunctive-disjunctive graph. We present both of these models next.

5.1.1 Integer Programming Models with Time Indexed Variables

Fisher [21] proposes one of the earliest pure integer programming models for the job shop scheduling problem. He has shown that there exists at least one optimal schedule so that all starting times are integer if the processing times for all the jobs are integers. Therefore, given T as an integer upper bound of time so that all jobs have to be

completed no later than T, the number of feasible schedules for any job i is bounded and finite. Let this number be denoted by Q_i. His formulation is as follows.

Let N be the number of jobs, M be the number of machines, and T be the time horizon, and p_{ij} be the processing time for job i on machine j. Suppose that C_i^q represents the completion time of job i under schedule q and α_{iqjt} represents the amount of machine j used during time period t by job i under schedule q. Also, let $c_{iq} = g_i\left(C_i^q\right)$ represent the cost of executing schedule q for job i, $\forall i = 1,2,...,N$, $q = 1,2,...,Q_i$. Then, for the objective of minimizing the total cost, we have the following formulation (see Fisher [21]).

Minimize $\sum_{iq} c_{iq} x_{iq}$ (5.1a)

subject to

$$\sum_{q} x_{iq} = 1, \ \forall i = 1,2,...,N \tag{5.1b}$$

$$\sum_{i,q} \alpha_{iqjt} x_{iq} \le R_{jt} \ \forall j = 1,2,...,M \ , t = 1,2,...,T \tag{5.1c}$$

$$x_{iq} \in \{0,1\} \ \forall i = 1,2,...,N \ , q = 1,2,...,Q_i . \tag{5.1d}$$

A binary variable $x_{iq} = 1$ if job i uses the schedule q, and $x_{iq} = 0$, otherwise. Constraint set (5.1b) ensures that only one schedule can be used for any job i, $\forall i = 1,2,...,N$. Constraint set (5.1c) is the machine capacity constraint so that the total amount required for any machine (resource) j cannot be more than the available capacity R_{jt} unit in any time period t, $\forall t = 1,2,...,T$. Note that α_{iqjt} is either 1 or 0, since job i under schedule q can either require machine j during time period t or not.

To solve this formulation, Fisher [21] proposes a branch and bound algorithm with a Lagrangian relaxation method, in which the constraint set (5.1c) is relaxed by multiplying a set of Lagrangian multipliers and added in the objective function as a penalty. The resulting problem becomes a set-covering problem.

Note that, c_{iq} is a function of the completion time of the jobs. As a result, we can express several job completion time related objectives for c_{iq}. These include, among others, total completion time ($\sum_{iq} C_i^q x_{iq}$), maximum completion time ($\max_{iq}\left\{C_i^q x_{iq}\right\}$), and makespan ($C_{\max}$), which we can represent as follows:

$C_{\max} \geq C_i^q \cdot x_{iq}$, $\forall i = 1,2,...,N$, $q = 1,2,...,Q_i$,

If d_i is the due date for job i, $\forall i = 1,2,...,N$, then we can express the due date related objectives as follows:

Lateness: $L_{iq} = C_i^q - d_i$, $\forall i = 1,2,...,N$, $q = 1,2,...,Q_i$;

Total lateness: $\sum_{iq} L_{iq} x_{iq}$;

Weighted lateness: $\sum_{iq} w_i L_{iq} x_{iq}$;

Maximum tardiness: $\max_{iq}\left\{\max\left(L_{iq},0\right)\cdot x_{iq}\right\}$.

The above definitions can be easily extended to the objectives of minimizing maximum lateness, maximum tardiness and their weighted counterparts. Moreover, if we define

$$U_{iq} = \begin{cases} 1, \text{if } C_i^q > d_i, \\ 0, \text{otherwise,} \end{cases}$$

we can also express the objectives of minimizing the total number of tardy jobs ($\sum_{iq} U_{iq} x_{iq}$) and the total weighted number of tardy job ($\sum_{iq} w_i U_{iq} x_{iq}$).

Both Chen *et al.* [16] and Chen and Hsia [17] consider the total weighted lateness, makespan, and the maximum tardiness as objective functions while using the same constraint sets as those in Fisher [21] for the scheduling problem encountered at the sorting and testing steps of the backend of a wafer fabrication facility. Moreover, Chen and Hsia [17] include the precedence constraints by adding the following constraint set (5.2) if job i must be performed immediately before job j, and job j requires machine k.

$C_i^q + 1 \le C_j^q - p_{jk}$, $\forall i, j = 1,2,...,N$, $i \ne j$ and job i immediately before job j .(5.2)

Chen *et al.* [16] consider the preemptiveness by altering the available capacities of machines (resources), R_{kt}, if the time period over which a machine breaks down is known. Both Chen *et al.* [16] and Chen and Hsia [17] use the same solution method as that in Fisher [21]. Note that the number of variables and constraints in this model can grow rather quickly with decrement in time grid.

5.1.2 Conjunctive-disjunctive Graph Models

Suppose that we have two jobs, jobs i and k that require processing times of p_i and p_k, respectively, on a machine. In order to prevent these two jobs from occupying the same machine at the same time, we must have $B_i - B_k \ge p_k$, or $B_k - B_i \ge p_i$, where B_i and B_k are the beginning times of jobs i and k, respectively.

In view of this noninterference restriction, Manne [48] proposed the first mathematical formulation of the job shop scheduling problem by introducing a set of binary variables y_{ik} to represent the noninterference restriction as follows.

$$(T + p_k)\cdot y_{ik} + (B_i - B_k) \ge p_k, \tag{5.3a}$$

$$(T + p_i)\cdot (1 - y_{ik}) + (B_k - B_i) \ge p_i, \tag{5.3b}$$

where T is a upper bound of $|B_i - B_k|$.

Balas [6] proposed a conjunctive-disjunctive graph to represent the noninterference requirement mentioned above. A conjunctive-disjunctive graph can be defined as follows. Suppose $\mathbf{D}$ represents a conjunctive-disjunctive graph $\mathbf{D} = (\mathbf{N}, \mathbf{Z} \cup \mathbf{W})$, where $\mathbf{N}$ is the set of nodes, $\mathbf{Z}$ is the set of conjunctive arcs and $\mathbf{W}$ is the set of disjunctive arcs. A node $j \in \mathbf{N}$ of this graph represents a job including nodes 0 and (n+1), which stand for a source and a sink of $\mathbf{D}$, respectively. A conjunctive arc $(i, j) \in \mathbf{Z}$ enforces a strict precedence between nodes i and j, that is, a node j can be started only after the completion of node i. A disjunctive pair of arcs $(i, j) \in \mathbf{W}$ and $(j, i) \in \mathbf{W}$, on the other hand, indicate that either node i is processed before node j, or vice versa. An example of a conjunctive-disjunctive graph involving 3 jobs and 4 machines is shown in Figure 5-

1. This graphical representation captures precedence among the operations of a job and among those assigned to the same machine. Balas [6] also proposed an algorithm to find the mini-maximal path (also termed critical path) of a conjunctive-disjunctive graph (**D**) in a finite number of steps. A simple description of the algorithm is as follows. For a typical iteration, the algorithm chooses a critical path from current graph and identifies a set of free arcs as candidates to be on the critical path. Note that only disjunctive arcs can be candidates for this step. Once a disjunctive arc is chosen to be on the critical path, then the complement of that arc is excluded and a new graph is formed. If no candidate is available, the algorithm backtracks. If all excluded arcs are backtracked, the algorithm terminates with the incumbent solution as optimal. Furthermore, he extended this conjunctive-disjunctive representation to formulate any constraints with and/or relations and coined the term, disjunctive programming, for this modeling procedure, in Balas [7].

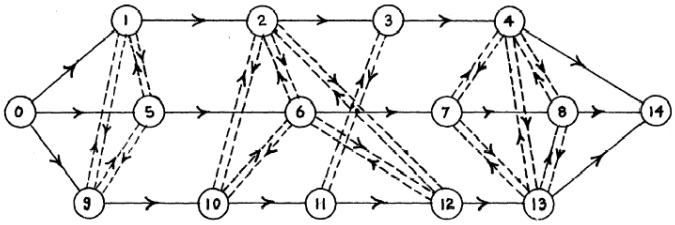

Figure 5-1 Conjunctive-disjunctive graph of a problem with 3 jobs and 4 machines [6]

Mascis and Pacciarelli [49] and Pacciarelli [55] propose a conjunctive-disjunctive graph with blocking and no-wait constraints, and designate the resulting graph as an alternative graph, for a job shop scheduling problem. Assume that a job j has to start within k time units after the completion of job i, as shown in the following Figure 5-2. If $k = 0$, then it is a tight no-wait constraint.

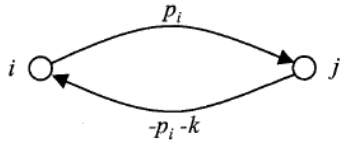

Figure 5-2 Graph representation for no-wait constraints [49]

Blocking constraints may arise in a job shop with finite buffer capacity between the machines. Suppose that job i is processed before job j and job i is blocking the processing of job j. Then, job j cannot be started until the processing of the successor

of job i, denoted as $\sigma(i)$, is started (see Figure 5-3). In Figure 5-3 (a), jobs i and j are blocking each other, and in Figure 5-3 (b), only job i is blocking. Pacciarelli [55] uses this alternative graph for the formulation of a job shop problem in a steel making plant.

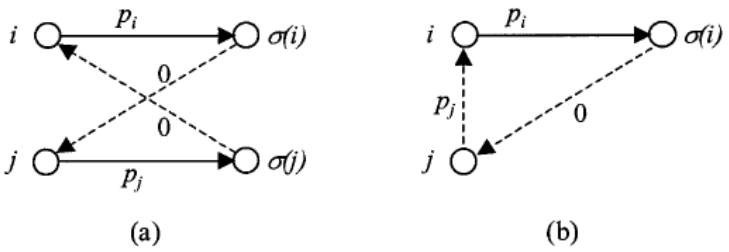

Figure 5-3 Graph representation for blocking constraints [49]

Here, we present the formulation for a job shop scheduling problem proposed by Applegate and Cook [3] that uses the conjunctive-disjunctive representation. Suppose that we have n jobs, $J = \{J_1, J_2, ..., J_n\}$, which are to be processed on m machines. For each job i, we are given a machine order $\sigma_i = \left(\sigma_1^i, \sigma_2^i, .., \sigma_{|J_i|}^i\right)$, which specifies the sequence in which job i is to be processed over the machines. The processing of a job i on machine j is designated by operation o_{ij}, $\forall i = 1,...,n,\ \ j = 1,...,m$. Each operation, o_{ij}, requires a processing time of p_{ij}. Let T_{ij} be the starting time of job i on machine j, and the problem is to determine a schedule of jobs on each machine so that the total time required to process all the jobs is minimized. The formulation, designated as CDgraph (conjunctive-disjunctive graph formulation), can be defined as follows.

CDgraph: Minimize $C_{\max}$ (5.4a)

subject to

$$C_{\max} \geq T_{i,\sigma_{|J_i|}^i} + p_{i,\sigma_{|J_i|}^i},\ \forall i = 1,...,n \tag{5.4b}$$

$$T_{i,\sigma_t^i} \geq T_{i,\sigma_{t-1}^i} + p_{i,\sigma_{t-1}^i},\ \forall i = 1,...,n, t = 2,...,|J_i| \tag{5.4c}$$

$$T_{ij} \geq T_{kj} + p_{kj} - K \cdot X_{ik}^j, \forall i,k = 1,...,n, i \neq k,\ j = 1,...,m \tag{5.4d}$$

$$T_{kj} \geq T_{ij} + p_{ij} - K \cdot (1 - X_{ik}^j), \forall i,k = 1,...,n, i \neq k,\ j = 1,...,m \tag{5.4e}$$

$$X_{ik}^j \in \{0,1\},\ \forall i,k = 1,...,n, i \neq k,\ j = 1,...,m \tag{5.4f}$$

$$T_{ij} \geq 0\,, \forall i = 1,...,n,\ j = 1,...,m\,. \tag{5.4g}$$

Note that the variable X_{ik}^{j} takes a value 1 if job i precedes job k on machine j; and 0, otherwise. Constraint set (5.4b) captures the time required to process all the jobs. Constraint set (5.4c) ensures that an operation of a job i, o_{i,σ_t^i}, cannot start processing until its immediately preceding operation, o_{i,σ_{t-1}^i}, has been completed. Constraint set (5.4d,e) states that, for any two jobs, job i and job k, which require the same machine j, either job k starts after job i has completed its operation on it, or vise versa. Note that, K can assume a value equal to a reasonable upper bound on $\left(T_{ij} + p_{ij} - T_{kj}\right)$, for all jobs $i,k = 1,...,n$, $i \neq k$ and all machines $j = 1,...,m$. Applegate and Cook [3] use a cutting-plane procedure for this formulation. In this cutting-plane scheme, all cuts require solution of a subproplem involving either complete enumeration or a simple heuristic.

5.2 A New Formulation for the Job Shop Scheduling Problem

Let $J_1, J_2,..., J_n$ be an ordered set of operations required for jobs $1,...,n$, respectively, and let $J = \{J_1, J_2,..., J_n\}$. Each operation of a job requires processing on one machine. Furthermore, we need to define the following sets. Let $M_1, M_2,..., M_m$ represent the subsets of operations that require the machines $1,2,...,m$, respectively, for their processing. By definition, $J_r \cap J_s = \phi$, $\forall r, s = \{1,...,n\}, r \neq s$ and $\bigcup_{r=1}^{n} J_r = J$. Also, each operation has to be processed on one and only one machine; that is $M_v \cap M_w = \phi$, $\forall v, w = \{1,...,m\}, v \neq w$ and $\bigcup_{v=1}^{n} M_v = J$. Note that, for any two operations i and j, we use $(i) \prec (j)$ to indicate the desired precedence relationship between operation i and operation j if they belong to the same job.

We define the following variables. Let,

$$x_{ij} = \begin{cases} 1\,, \text{ if operation } i \text{ directly precedes operation } j \text{ on a machine,} \\ 0\,, \text{ otherwise, } \forall i, j \in J \cup \{0\},\ i \neq j\,, \end{cases}$$

and

$$y_{ij} = \begin{cases} 1 \text{, if operation } i \text{ precedes operations } j \text{ on a machine,} \\ 0 \text{, otherwise, } \forall i, j \in J \text{, } i \neq j \text{.} \end{cases}$$

The $\boldsymbol{x}$-variables and $\boldsymbol{y}$-variables can only have a value of 1 for the two operations i and j that require the same machine. Moreover, we use a dummy starting operation 0, which has connotations of the base city in ATSP. The $\boldsymbol{x}$-variables define a Hamiltonian circuit, whereas the $\boldsymbol{y}$-variables, capture the general precedence among the jobs that are processed on a machine. Hence, the indices for the $\boldsymbol{y}$-variables do not include the starting dummy operation 0. We also define a time-based variable C_i to represent the completion time of operation i. Our formulation for the job shop scheduling problem is as follows:

JSCD0: Minimize $C_{\max}$ (5.5a)

subject to

$$C_{\max} \geq C_i, \ \forall i \in J \tag{5.5b}$$

$$C_j \geq C_i + p_j, \ \forall i, j \in J_r \ , r = 1,...,n \text{, and } (i) \prec (j) \tag{5.5c}$$

$$\sum_{j \in M_v, j \neq i} x_{iv} + x_{i0} = 1, \ \forall i \in M_v \tag{5.5d}$$

$$x_{0j} + \sum_{i \in M_v, i \neq j} x_{ij} = 1, \forall j \in M_v \tag{5.5e}$$

$$\sum_{i \in M_v} x_{i0} = 1, \ \forall v = 1,...,m \tag{5.5f}$$

$$\sum_{i \in M_v} x_{0i} = 1, \ \forall v = 1,...,m \tag{5.5g}$$

$$y_{ij} \geq x_{ij}, \forall i, j \in M_v, i \neq j \text{, and} \left|M_v\right| \geq 2 \text{, } v = 1,...,m \tag{5.5h}$$

$$y_{ij} + y_{ji} = 1, \forall i, j \in M_v, i \neq j \text{, and} \left|M_v\right| \geq 2 \text{, } v = 1,...,m \tag{5.5i}$$

$$-(1 - x_{ik}) \leq (y_{ij} - y_{kj}) \leq (1 - x_{ik}), \ \forall i, j, k \in M_v \text{, } i \neq j \neq k \text{, and} \left|M_v\right| \geq 3 \text{, } v = 1,...,m \tag{5.5j}$$

$$y_{ij} \geq x_{0i}, \forall i, j \in M_v, i \neq j \text{, and} \left|M_v\right| \geq 2 \text{, } v = 1,...,m \tag{5.5k}$$

$$y_{ji} \geq x_{i0}, \forall i, j \in M_v, i \neq j, \text{ and } |M_v| \geq 2, v = 1,...,m \qquad (5.5\ell)$$

$$C_j \geq C_i + p_j - K \cdot (1 - y_{ij}), \forall i, j \in M_v, i \neq j, \text{ and } |M_v| \geq 2, v = 1,...,m \qquad (5.5\text{m})$$

$$C_i \geq C_j + p_i - K \cdot y_{ij}, \forall i, j \in M_v, i \neq j, \text{ and } |M_v| \geq 2, v = 1,...,m \qquad (5.5\text{n})$$

$$C_i \geq 0, \forall i \in J, \; x_{ij} \in \{0,1\}, \forall i, j \in J \cup \{0\}, \; i \neq j, \; y_{ij} \geq 0, \; \forall i, j \in J, \; i \neq j. \qquad (5.5\text{o})$$

In JSCD0, constraint set (5.5b) captures the completion time of all the jobs. Constraint set (5.5c) ensures that the earliest starting time of operation j of a job r has to be processed only after the completion of its previous operation i. The constraint sets (5.5d,e) are the standard assignment constraints and indicate that, for a given machine, an operation should be succeeded by and preceded by, respectively, only one operation (including the dummy operation) among all the operations that require that machine. The constraint sets (5.5f,g) enforce that every sequence on a machine starts from and ends at the dummy operation. (5.5h,k,ℓ) logically relate the precedence variables $\boldsymbol{y}$ to the immediate precedence variables $\boldsymbol{x}$; (5.5i) asserts that either an operation i precedes operation j or vice versa, $\forall i, j \in M_v, i \neq j$, and (5.5j) enforces that if $x_{ik} = 1$ for any $i \neq k$, then for any other operation j, we must have $y_{ij} = y_{kj}$ (whether this pair equals 0 or 1). Observe that due to (5.5i), this also, then, ensures that $y_{ji} = y_{jk}$. Finally, (5.5m,n) are the disjunctive constraints.

First, we establish the validity of the model JSCD0 given by (5.5) above.

Proposition 5.1. The formulation JSCD0 given by (5.5) correctly models the job shop scheduling problem.

Proof. For a job shop problem, we can determine a schedule if, for each machine, we are given the sequence to process the operations assigned to that machine. A feasible sequence of operations for a machine can be treated here as a feasible ATSP tour that starts from and ends at a dummy operation 0. Therefore, JSCD0 is a valid formulation if the $\boldsymbol{x}$ solution, which defines these ATSP tours, avoids any subtours.

We prove this by contradiction. It essentially follows the line of arguments used in the proof of Proposition 3.1. Given any feasible solution to JSCD0, suppose that, for some machine v with at least two operations to be processed on ($|M_v| \geq 2$), the

assignment solution $\boldsymbol{x}$ admits a subtour on some operations i ($i \in M_v, v = 1,...,m$). Let this subtour be $p_1, p_2, ..., p_t, p_1$. Note that all these operations are assigned to the same machine v ($\forall p_1, p_2, ..., p_t \in M_v$). Hence, because $x_{p_1 p_2} = x_{p_2 p_3} = ... = x_{p_{t-2}, p_{t-1}} = 1$, we get by using each of these in (5.5j) that,

$$y_{p_1 p_t} = y_{p_2 p_t} = y_{p_3 p_t} = ... = y_{p_{t-1}, p_t} \quad . \tag{5.6}$$

Moreover, $x_{p_{t-1}, p_t} = 1$ would imply from (5.5h) that $y_{p_{t-1}, p_t} = 1$. Consequently, $y_{p_1 p_t} = 1$ from (5.6). But $x_{p_t p_1} = 1$ also implies that $y_{p_t p_1} = 1$, see (5.5h), which contradicts (5.5i). This completes the proof. □

5.3 Implementation of JSCD0 on the Job Shop Scheduling Problems from the Literature

First, we use the $6 \times 6 \times 6$ test problem (see Table 5-1) given in Muth and Thompson [53] to show how to relate an optimal solution by solving JSCD0 to an optimal job shop schedule. Table 5-1 includes the processing time (p) and required machine (σ) for all operations (Op).

Table 5-1 Job profile for a 6×6×6 test problem [53]

Op	Job1		Job2		Job3		Job4		Job5		Job6	
	p	σ	p	σ	p	σ	p	σ	p	σ	p	σ
1	1	3	8	2	5	3	5	2	9	3	3	2
2	3	1	5	3	4	4	5	1	3	2	3	4
3	6	2	10	5	8	6	5	3	5	5	9	6
4	7	4	10	6	9	1	3	4	4	6	10	1
5	3	6	10	1	1	2	8	5	3	1	4	5
6	6	5	4	4	7	5	9	6	1	4	1	3

The LP relaxation of JSCD0 gives a lower bound of 47 on the optimal $C_{\max}$ value in 0.16 cpu seconds. The optimal $C_{\max}$ value is 55, obtained in 585.2 seconds. The runs were made on a Dell PC with 664MHz CPU Pentium III processors and 512MB RAM, running the Windows XP operating system, and we used AMPL (version 8.1) along with CPLEX MIP Solver (version 9.0). The operation sequence on each machine for the optimal solution is shown in Table 5-2. The Gantt chart is shown in Figure 5-4.

Table 5-2 The optimal operation sequence on each machine

Machine	Sequence (Job, Order)
1	(Job1,2) → (Job4,2) → (Job3,4) → (Job6,4) → (Job2,5) →(Job5,5)
2	(Job2,1) → (Job4,1) → (Job6,1) → (Job5,2) → (Job1,3) →(Job3,5)
3	(Job1,1) → (Job3,1) → (Job2,2) → (Job5,1) → (Job4,3) →(Job6,6)
4	(Job3,2) → (Job6,2) → (Job4,4) → (Job1,4) → (Job2,6) →(Job5,6)
5	(Job2,3) → (Job5,3) → (Job4,5) → (Job3,6) → (Job6,5) →(Job1,6)
6	(Job3,3) → (Job6,3) → (Job2,4) → (Job1,5) → (Job5,4) →(Job4,6)

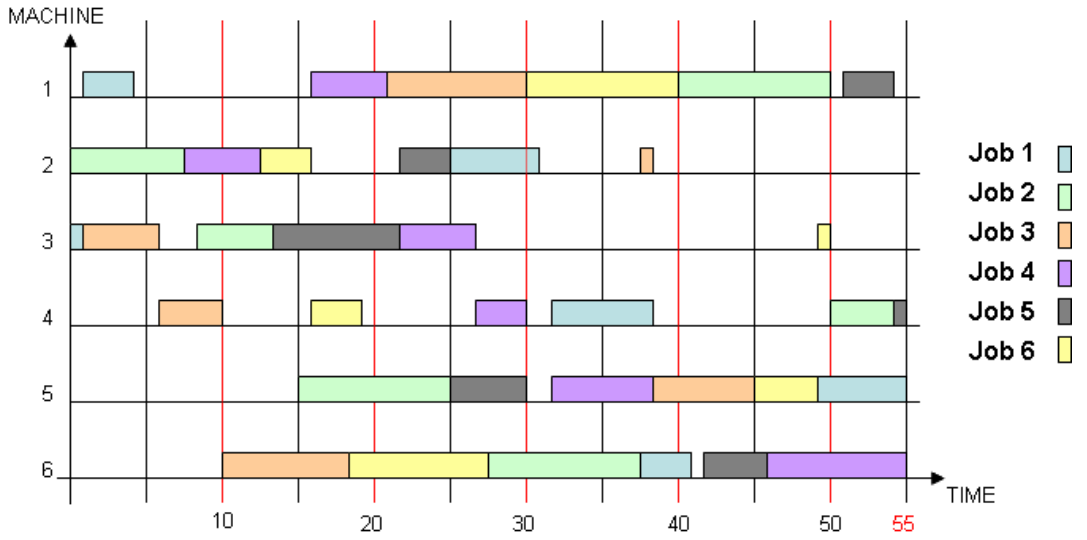

Figure 5-4 The Gantt chart for the optimal operation sequence on each machine

Next, we use the 10×10×10 test problem (see Table 5-2) given in Muth and Thompson [53] to compare the lower bound values obtained by different formulations for the job shop scheduling problem. Table 5-2 presents the processing time (p) and required machine (σ) for all operations (Op), as well as the total processing time for each job ($\sum$) for this problem. The various formulations that are used include those obtained by augmenting different tight ATSP constraints to the JSCD0 formulation. Also the lower bound values obtained by preempt heuristic and 1-machine heuristic procedures (see Applegate and Cook [3]) are shown in Table 5-4.

Table 5-4 shows the lower bound values (z_{LP}) and the cpu time (in seconds) required to obtain the optimal LP solution. Note that, we also include the following constraint set (5.7) on machine capacity to further tighten the formulation JSCD0.

$$C_j \geq \sum_{i \in M_k, i \neq j} p_i \cdot y_{ij} + p_j, \ \forall i, j \in M_k, i \neq j, \text{ and } |M_k| \geq 2, k = \{1,...,m\}. \qquad (5.7)$$

Table 5-3 Job profile for a 10×10×10 test problem [53]

Op	Job1		Job2		Job3		Job4		Job5		Job6		Job 7		Job 8		Job9		Job10	
	p	σ	p	σ	p	σ	p	σ	p	σ	p	σ	p	σ	p	σ	p	σ	p	σ
1	29	1	43	1	91	2	81	2	14	3	84	3	46	2	31	3	76	1	85	2
2	78	2	90	3	85	1	95	3	6	1	2	2	37	1	86	1	69	2	13	1
3	9	3	75	5	39	4	71	1	22	2	52	6	61	4	46	2	76	4	61	3
4	36	4	11	10	74	3	99	5	61	6	95	4	13	3	74	6	51	6	7	7
5	49	5	69	4	90	9	9	7	26	4	48	9	32	7	32	5	85	3	64	9
6	11	6	28	2	10	6	52	9	69	5	72	10	21	6	88	7	11	10	76	10
7	62	7	46	7	12	8	85	8	21	9	47	1	32	10	19	9	40	7	47	6
8	56	8	46	6	89	7	98	4	49	8	65	7	89	9	48	10	89	8	52	4
9	44	9	72	8	45	10	22	10	72	10	6	5	30	8	36	8	26	5	90	5
10	21	10	30	9	33	5	43	6	53	7	25	8	55	5	79	4	74	9	45	8
Σ	395		510		568		655		393		496		416		539		597		540	

Table 5-4 Job profile for the 10×10×10 test problem

Problem	Formulation	Without (5.7)		With (5.7)	
		z_{LP}	CPU time (LP)	z_{LP}	CPU time (LP)
MT10 (10×10×10)	Preempt[3]	808	-	808	-
	1-Machine[3]	808	-	808	-
	JSCD0	655	0.34	729.70	0.69
	JSCD_ATSP0	655	0.42	729.70	0.91
	JSCD_ATSP2	655	9.64	729.70	10.17
	JSCD_ATSP3	655	16.53	729.70	17.91
	JSCD_ATSP5	655	27.92	729.70	29.71
	JSCD_ATSP6	655	30.03	729.70	33.14

In Table 5-4, both the '1-Machine' and 'preempt' lower bound procedures as described in Applegate and Cook [3] use the concepts of machine-based and job-based bounds. The '1-Machine' procedure does not allow preemption of an operation, while it is permitted in the 'Preempt' procedure. The JSCD_ATSP0 formulation is obtained by augmenting JSCD0 formulation with the following constraint sets:

$$y_{ij} \geq x_{1i},\ \forall i, j \in M_k, i \neq j, \text{ and } |M_k| \geq 2,\ k = \{1,...,m\}. \tag{5.8a}$$

$$y_{ji} \geq x_{i1},\ \forall i, j \in M_k, i \neq j, \text{ and } |M_k| \geq 2,\ k = \{1,...,m\}. \tag{5.8b}$$

The JSCD_ATSP2 formulation is obtained by augmenting a relaxed JSCD0 formulation, obtained by deleting constraint sets (5.5h) and (5.5j- ℓ), with the following constraint sets:

$$0 \leq f_{ij}^{v} \leq x_{iv},\ \forall i, j, v \in M_k,\ i \neq j \neq v, \text{ and } |M_k| \geq 3,\ k = \{1,...,m\}. \tag{5.9a}$$

$$\sum_{v=2, v \notin \{i,j\}}^{n} f_{ij}^{v} + x_{ij} = y_{ij},\ \forall i, j \in M_k, i \neq j, \text{ and } |M_k| \geq 3,\ k = \{1,...,m\}. \tag{5.9b}$$

$$x_{1v} + \sum_{i=2, i\notin\{v,j\}}^{n} f_{ij}^{v} = y_{vj},\ \forall i, j \in M_k, i \neq j, \text{ and } |M_k| \geq 3, k = \{1,...,m\}. \quad (5.9c)$$

In the JSCD_ATSP3 formulation is obtained by adding the following constraint sets to JSCD_ATSP2:

$$y_{ij} + y_{jk} + y_{ki} \leq 2,\ \forall i, j, v \in M_k,\ i \neq j \neq v, \text{ and } |M_k| \geq 3, k = \{1,...,m\}. \quad (5.10)$$

Similarly, the JSCD_ATSP5 formulation is obtained by adding the following constraint set to JSCD_ATSP2:

$$(y_{ij} + x_{ji}) + y_{jk} + y_{ki} \leq 2, \forall i, j, v \in M_k,\ i \neq j \neq v, \text{ and } |M_k| \geq 3, k = \{1,...,m\}. \quad (5.11a)$$

$$x_{1j} + x_{j1} \leq 1, \forall j \in M_k, \text{ and } |M_k| \geq 2, k = \{1,...,m\}. \quad (5.11b)$$

Finally, the JSCD_ATSP6 formulation is obtained by adding the constraint set (5.11a) to JSCD_ATSP2:

As indicated in Table 5-4, the lower bound values (z_{LP}) obtained by solving the LP relaxations of the variations JSCD0: JSCD_ATSP0, JSCD_ATSP2, JSCD_ATSP3, JSCD_ASP5, and JSCD_ATSP6 are identical to that obtained by solving the LP relaxation of JSCD0. The inclusion of constraint set (5.7) gives tighter lower bound values. However, these lower bound values are still not as tight as those obtained from the Preempt and 1-Machine heuristic procedures.

Moreover, the lower bound value obtained by solving the LP relaxation of all the variations of JSCD0 is the same as the maximum total processing time among the jobs. Fisher *et al.* [22] consider a formulation similar to JSCD0 and use a set of nonnegative weights to aggregate the disjunctive constraints, as in (5.5m,n). Note that, when the binary requirement on the $\boldsymbol{y}$-variables is relaxed, the lower bound value would be equal to the maximum total processing time among all the jobs (i.e. $LB = \max_j \left\{ \sum_{i \in J_j} p_i \right\}$).

Chapter 6 Research Contribution and Recommendations for Future Research

6.1 Research Contribution

In this dissertation, we have presented a new class of polynomial length formulations for the asymmetric traveling salesman problem (ATSP), which lies at the core of several scheduling problems that arise in the manufacturing and distribution logistics areas. We have used the Reformulation-Linearization Technique (RLT) [63,64,65] to lift an ordered path-based model with logical restrictions. We show that a relaxed version of this formulation is equivalent to a flow-based ATSP model, which in turn, is tighter than the formulation based on the exponential number of Dantzig-Fulkerson-Johnson (DFJ) subtour elimination constraints. The proposed lifting idea is applied to derive a variety of new formulations for the ATSP and designated as ATSP1, ATSP2, ATSP2R, ATSP2R$^-$ (which is equivalent the tightened version, ATSP-FL2, of ATSP-FL, see Proposition 3.5), ATSP3, ATSP4, ATSP5, ATSP6, ATSP7, ATSP8, and ATSP9. A detailed analysis of these formulations is carried out (see Section 3.3) to show that some of these formulations are the tightest among those presented in the literature.

Computational results are presented in Section 3.4 to exhibit the relative tightness of our formulations and the efficacy of the proposed lifting process. The results show that the lower bounds (z_{LP}) obtained via the LP relaxations of ATSP1, ATSP2, ATSP2R, ATSP2R$^-$, ATSP3, ATSP5, ATSP6, ATSP8 and ATSP9 are at least as tight as those obtained by solving the LP relaxations of ATSP-FL and ATSP-FL2. Furthermore, ATSP5 and ATSP6 produce the tightest lower bounds, and ATSP3 yields the second tightest lower bound value over the test problems used.

While the computational results demonstrate the efficacy of employing the proposed theoretical RLT and logical lifting ideas, yet it remains of practical interest to take due advantage of the tightest formulations. We, then, attempt to solve the LP relaxation of

our tightest formulation, ATSP6, to (near-) optimality using a deflected subgradient algorithm with average direction strategy (SA_ADS) (see see Sherali and Ulular [69]) as well as two nondifferentiable optimization (NDO) methods, namely, the variable target value method (VTVM) presented in Sherali *et al*. [66] and the trust region target value method (TRTV) presented in Lim and Sherali [46], on the Lagrangian dual formulation of the associated problem. Two test problems adapted from the TSP library, namely br4 and br17, are used to evaluate the above algorithm. The subgradient method, SA_ADS, effectively solves the test problem br4 close to optimality, but it fails to solve the test problem br17 to (near-) optimality. For the NDO methods, the near optimal values obtained by the VTVM, using the canonical format of the LP relaxation of ATSP6, generates the tightest value under the same parameter settings.

Another approach is to derive a set of strong valid inequalities based on these tighter formulations through a suitable surrogation process, and to use these within a compact and manageable formulation. We suggest such a procedure to derive a set of strong valid inequalities. Our computational results show that, when the dual optimal solution is available, the associated strong valid inequalities generated from our procedure can successfully lift the LP relaxation of a less tight formulation, such as ATSP2R^{-}, to be as tight as the tightest formulation, such as ATSP6.

The presence of precedence constraints within the ATSP framework is encountered quite often in practice and our flow-based ATSP formulation can very conveniently capture these precedence constraints. We also present computational results to depict the tightness of our precedence-constrained ATSP (PCATSP) formulation. The results reveal that PCATSP5 and PCATSP6 again provide the tightest lower bounds (z_{LP}), and that, the lower bound values generated by PCATSP3 are also close to those obtained by PCATSP5 and PCATSP6. The LP-IP gap is typically less than 1.9%, except for Problem br17.10 where the gap is 19.2%.

We, then, apply our formulations to the hot strip rolling scheduling problem, which involves the processing of hot steel slabs, in a pre-specified precedence order, on one or more rollers. The single-roller hot strip rolling scheduling problem can be directly formulated as a PCATSP. Then, we develop new formulations for the multiple-

asymmetric traveling salesman problem (mATSP) and extend these formulations to include a pre-specified, special type of precedence order in which to process the slabs. These resulting formulations are designated as formulations for restricted precedence-constrained multiple-asymmetric traveling salesman problem (rPCmATSP). In Section 4.4, we design a series of experiments and present computational results by solving our formulations using the default branch and bound method in CPLEX as well as the Benders' decomposition [13] and Modified Benders´ methods. We further introduce an adjustable density ratio of the associated precedence network and use randomly generated test problems for the study of the effect of various density ratios in solving these scheduling problems. We use a set of five randomly generated problems including 30 slabs. Furthermore, we apply 8 different network densities on these test problems to study the impact of network density on computational performance. The network densities used are 90.25% (p=0.05), 81% (p=0.1), 72.25% (p=0.15), 64% (p=0.2), 56.25% (p=0.25), 20.25% (p=0.55), 9% (p=0.70), and 1% (p=0.9). The results indicate that both Benders´ decomposition and Modified Benders´ methods can solve most of these test problems within 10 cpu hours. Moreover, the results show that when the network densities are 90.25% (p=0.05), 81% (p=0.1), and 1% (p=0.9), all five test problems can be solved within 1,000 seconds. When the network density is 56.25% (p=0.25) or 20.25% (p=0.55), it takes longer to solve the associated test problems.

In Chapter 5, we continue our discussion of using conjunctive-disjunctive formulation for the job shop scheduling problem. We provide a compact formulation for the job shop scheduling problem, designated as JSCD (job shop conjunctive-disjunctive) formulation and, then, we use two test problems, 6×6×6 and 10×10×10, given in Muth and Thompson [53] to demonstrate the optimal schedule and the lower bound values obtained by solving the LP relaxation of various formulations. However, we observe that the lower bound values obtained by solving the LP relaxation of all the variations of JSCD formulation are the same, which equals to the maximum total processing time among all jobs (i.e. $LB = \max_j \left\{ \sum_{i \in J_j} p_i \right\}$), as indicated in Fisher *et al.* [22].

6.2 Recommendations for Future Research

For future research, we recommend two directions: development of efficient algorithms for solving the LP relaxations of our new formulations as well as for the underlying integer programs.

The development of efficient solution procedures for the LP relaxations can be crucial for taking advantage of our tightest formulations (PC)ATSP5 and (PC)ATSP6. Our preliminary results show that the near optimal values obtained by the VTVM, a nondifferentiable optimization method, are very close to the target optimal values. It would be worthwhile to study the special structure of the LD reformulations of the LP relaxations for our formulations and develop non-linear algorithms to generate (near-) optimal solutions by solving the resulting formulations. If such an algorithm is available, the corresponding dual solution for the (near-) optimal LP solution can, then, be used to derive strong valid inequalities through a suitable surrogation process, and hence, can be used to further tighten the more compact and manageable formulations.

From the computational experiments in Chapter 4, we have found that both Benders´ decomposition and Modified Benders´ methods dominate CPLEX in obtaining optimal solutions. It is, therefore, viable to develop a special and efficient branch and bound scheme to obtain an optimal IP solution.

Appendix I. Computational Comparison of Various ATSP Formulations

Table A-1 displays the results obtained pertaining to the lower bound value (z_{LP}) derived by solving the LP relaxation, the cpu time (in seconds) required to compute the optimal LP solution, the percentage gap between the LP-based lower bound and the optimal integer solution, z_{IP} , the additional cpu time (in seconds) required to obtain an optimal solution (limited to 10,000 cpu seconds), and the number of branching nodes. Note that the number in bold indicates the highest z_{LP} value obtained among all the models for each test case.

Table A- 1 Computational Results for the Various ATSP Formulations

Problem	Formulation	z_{LP}	CPU time (LP)	Gap (%)	Best Integer	CPU time (IP)	Nodes
br17 (z_{IP} =39)	ATSP-SSB2	28.00	0.16	28.2	39	3,241.5	71,572
	ATSP-SD	27.68	0.05	29.0	39	143.0	23,495
	ATSP-GP	18.00	0.03	53.8	39	10,000*	265,137
	ATSP-GP1	22.00	0.02	43.6	39	1,441.1	204,363
	ATSP-GP2	28.00	0.06	28.2	39	967.3	58,026
	ATSP-GP3	28.00	0.15	28.2	39	3,245.5	37,728
	ATSP-FL	**39.00**	1.63	0.0	39	16.9	0
	ATSP0	22.00	0.22	43.6	39	9,106.4	183,737
	ATSP1	**39.00**	19.56	0.0	39	86.3	0
	ATSP2	**39.00**	5.81	0.0	39	2.3	0
	ATSP2R	**39.00**	0.50	0.0	39	2.0	0
	ATSP2R$^-$	**39.00**	1.67	0.0	39	13.2	0
	ATSP3	**39.00**	10.55	0.0	39	57.6	0
	ATSP4	22.00	0.36	43.6	39	10,000*	220,641
	ATSP5	**39.00**	12.98	0.0	39	55.7	0
	ATSP6	**39.00**	10.88	0.0	39	51.7	0
	ATSP7	28.00	0.17	28.2	39	3,166.3	51,091
	ATSP8	**39.00**	14.91	0.0	39	101.5	0
	ATSP9	**39.00**	12.66	0.0	39	66.0	0
ftv33 (z_{IP} =1286)	ATSP-SSB2	**1286.00**	17.03	0.0	1286	5.1	0
	ATSP-SD	1224.50	0.42	4.8	1286	473.7	13,449
	ATSP-GP	1224.68	1.81	4.8	1286	36.2	4
	ATSP-GP1	1226.25	3.08	4.6	1286	62.7	2
	ATSP-GP2	**1286.00**	3.83	0.0	1286	4.5	0
	ATSP-GP3	**1286.00**	13.45	0.0	1286	12.3	0
	ATSP-FL	**1286.00**	49.53	0.0	1286	57.0	0
	ATSP0	1224.68	23.67	4.8	1362	10,000*	578
	ATSP1	**1286.00**	16,536.60	0.0	1286	1,662.2	0
	ATSP2	**1286.00**	248.87	0.0	1286	196.53	0
	ATSP2R	**1286.00**	34.70	0.0	1286	40.51	0
	ATSP2R$^-$	**1286.00**	27.55	0.0	1286	35.5	0
	ATSP3	**1286.00**	7,147.47	0.0	1286	940.5	0
	ATSP4	1229.08	43.78	4.4	1286	1,636.3	5
	ATSP5	**1286.00**	8,095.53	0.0	1286	1,292.0	0
	ATSP6	**1286.00**	5,540.85	0.0	1286	1,284.5	0
	ATSP7	**1286.00**	9.09	0.0	1286	11.2	0
	ATSP8	**1286.00**	26,271.20	0.0	1286	0	0
	ATSP9	**1286.00**	6,912.45	0.0	1286	646.0	0
ftv35 (z_{IP} =1473)	ATSP-SSB2	1456.89	26.77	1.1	1473	514.3	17
	ATSP-SD	1415.51	0.34	3.9	1473	1,754.3	27,331
	ATSP-GP	1422.86	2.25	3.4	1473	406.2	165
	ATSP-GP1	1426.90	4.91	3.1	1473	479.7	213
	ATSP-GP2	1456.06	5.55	1.2	1473	133.0	9
	ATSP-GP3	1456.69	14.14	1.1	1473	1,289.2	69
	ATSP-FL	1457.33	81.01	1.1	1473	477.2	10
	ATSP0	1424.57	23.95	3.3	1473	6,364.8	405
	ATSP1	1460.20	48,198.30	0.9	1710	10,000*	0
	ATSP2	1460.20	1,448.53	0.9	2040	10,000*	0
	ATSP2R	1457.33	61.02	1.1	1473	380.9	6
	ATSP2R$^-$	1457.33	86.12	1.1	1473	1,024.1	43
	ATSP3	1460.20	10,876.10	0.9	1885	10,000*	0
	ATSP4	1427.00	44.14	3.1	1475	10,000*	311
	ATSP5	**1463.41**	10,588.00	0.7	2036	10,000*	0
	ATSP6	**1463.41**	19,271.70	0.7	1806	10,000*	0
	ATSP7	1456.89	8.91	1.1	1473	973.8	51
	ATSP8	1460.20	13,839.90	0.9	1974	10,000*	0
	ATSP9	1460.20	15,583.00	0.9	1995	10,000*	0

Table A-1 Computational Results for the Various ATSP Formulations (con't)

Problem	Formulation	z_{LP}	CPU time (LP)	Gap (%)	Best Integer	CPU time (IP)	Nodes
ftv38 (z_{IP} =1503)	ATSP-SSB2	1484.00	44.85	1.3	1503	1,360.2	41
	ATSP-SD	1458.22	0.78	3.0	1503	1,053.8	11,484
	ATSP-GP	1459.50	3.63	2.9	1503	463.4	131
	ATSP-GP1	1461.08	6.66	2.8	1503	609.5	147
	ATSP-GP2	1481.29	6.81	1.4	1503	528.7	97
	ATSP-GP3	1483.00	28.26	1.3	1503	2,216.4	61
	ATSP-FL	1482.00	100.12	1.4	1503	1,440.7	40
	ATSP0	1459.50	41.89	2.9	1503	6,887.8	164
	ATSP1	1486.10	47,445.70	1.1	-**	10,000*	0
	ATSP2	1486.10	14,426.60	1.1	2309	10,000*	0
	ATSP2R	1482.00	239.13	1.4	1503	1,776.7	22
	ATSP2R$^-$	1482.00	110.61	1.4	1503	540.2	18
	ATSP3	1486.10	39,336.40	1.1	-**	10,000*	0
	ATSP4	1463.50	92.42	2.6	1503	10,000*	98
	ATSP5	**1488.22**	23,127.40	1.0	1795	10,000*	0
	ATSP6	**1488.22**	8,953.23	1.0	-**	10,000*	0
	ATSP7	1484.00	57.89	1.3	1503	2,269.9	90
	ATSP8	1486.10	10,630.70	1.1	1901	10,000*	0
	ATSP9	1486.10	30,898.60	1.1	-**	10,000*	0
p43 (z_{IP} =5620)	ATSP-SSB2	241.22	123.58	95.7	5663	10,000*	111
	ATSP-SD	864.58	1.49	84.6	5635	10,000*	103,445
	ATSP-GP	184.00	3.17	96.7	5662	10,000*	1,091
	ATSP-GP1	216.00	9.75	96.2	5630	10,000*	2,362
	ATSP-GP2	241.14	7.06	95.7	5635	10,000*	969
	ATSP-GP3	242.00	45.50	95.7	5648	10,000*	246
	ATSP-FL	**5611.00**	2,392.55	0.2	5639	10,000*	0
	ATSP0	216.00	148.96	96.2	5682	10,000*	33
	ATSP1	**5611.00**	162,993.00	0.2	-**	10,000*	0
	ATSP2	**5611.00**	17,814.10	0.2	-**	10,000*	0
	ATSP2R	**5611.00**	1,173.61	0.2	5648	10,000*	0
	ATSP2R$^-$	**5611.00**	1,138.02	0.2	5633	10,000*	0
	ATSP3	**5611.00**	30,629.90	0.2	-**	10,000*	0
	ATSP4	216.00	233.89	96.2	5647	10,000*	17
	ATSP5	**5611.00**	44,350.30	0.2	-**	10,000*	0
	ATSP6	**5611.00**	35,193.60	0.2	-**	10,000*	0
	ATSP7	241.22	121.10	95.7	5641	10,000*	71
	ATSP8	**5611.00**	130,264.00	0.2	-**	10,000*	0
	ATSP9	**5611.00**	50,915.40	0.2	-**	10,000*	0
ftv44 (z_{IP} =1613)	ATSP-SSB2	1584.88	136.65	1.8	1613	10,000*	32
	ATSP-SD	1573.75	0.14	2.4	1613	511.9	2,728
	ATSP-GP	1578.83	10.31	2.1	1613	3,489.3	332
	ATSP-GP1	1582.00	17.73	1.9	1613	689.7	75
	ATSP-GP2	1584.88	14.50	1.8	1613	847.0	42
	ATSP-GP3	1584.88	68.64	1.8	1613	4,564.2	49
	ATSP-FL	1584.88	262.13	1.7	1613	6,107.1	78
	ATSP0	1580.17	316.18	2.0	1623	10,000*	30
	ATSP1	1590.74	158,416.00	1.4	-**	10,000*	0
	ATSP2	1590.74	42,928.20	1.4	2038	10,000*	0
	ATSP2R	1584.88	1,528.93	1.7	1677	10,000*	14
	ATSP2R$^-$	1584.88	277.81	1.7	1613	5,966.3	47
	ATSP3	1590.91	109,782.00	1.4	-**	10,000*	0
	ATSP4	1582.00	417.41	1.9	1811	10,000*	0
	ATSP5	**1594.94**	69,517.90	1.1	-**	10,000*	0
	ATSP6	**1594.94**	159,105.00	1.1	-**	10,000*	0
	ATSP7	1584.88	150.80	1.7	1691	10,000*	11
	ATSP8	1590.74	81,880.40	1.4	-**	10,000*	0
	ATSP9	1590.74	169,472.00	1.4	-**	10,000*	0

- * Exceeded time limit.
- ** No integer solution was obtained within the time limit.

Appendix II. Computational Comparison of Various PCATSP Formulations

Table A-2 displays the results obtained pertaining to the lower bound value (z_{LP}) derived by solving the LP relaxation, the cpu time (seconds) required to compute z_{LP}, the percentage gap between z_{LP} and optimal integer solution z_{IP}, the cpu time (seconds) required to obtain the best integer solution, and the number of branching nodes.

Table A- 2 Computational Results for the Various PCATSP Formulations

Problem	Formulation	z_{LP}	CPU time (LP)	Gap (%)	Best Integer	CPU time (IP)	Nodes
esc12	PCATSP-SSB1	1554.00	0.03	7.2	1675	1.94	199
(z_{IP} =1675)	PCATSP-SD	1520.35	0.02	9.2	1675	1.16	123
	PCATSP-GP	1507.50	0.02	10.0	1675	1.36	439
	PCATSP-GP1	1518.50	0.02	9.3	1675	0.95	126
	PCATSP-GP2	1528.50	0.03	8.7	1675	2.42	445
	PCATSP-GP3	1535.63	0.11	8.3	1675	4.88	137
	PCATSP0	1507.50	0.03	10.0	1675	2.60	125
	PCATSP1	1649.50	0.95	1.5	1675	4.62	0
	PCATSP2	1649.50	0.38	1.5	1675	1.67	0
	PCATSP2R	1649.50	0.06	1.5	1675	0.17	0
	PCATSP2R⁻	1612.00	0.09	3.8	1675	1.41	8
	PCATSP3	1649.50	0.36	1.5	1675	1.03	0
	PCATSP4	1566.40	0.05	6.5	1675	1.64	92
	PCATSP5	**1675.00**	0.45	0	1675	0.50	0
	PCATSP6	**1675.00**	0.48	0	1675	0.50	0
br17.10	PCATSP-SSB1	27.29	0.34	50.4	55	111.6	2,015
(z_{IP} =55)	PCATSP-SD	23.64	0.11	57.0	55	2,895.8	314,818
	PCATSP-GP	18.00	0.06	67.3	55	10,000*	268,768
	PCATSP-GP1	18.00	0.05	67.3	55	10,000*	263,831
	PCATSP-GP2	28.00	0.08	49.1	55	10,000*	248,393
	PCATSP-GP3	28.00	0.47	49.1	55	10,000*	56,546
	PCATSP0	22.33	0.13	59.4	55	4,451.9	126,604
	PCATSP1	39.70	12.23	27.8	55	10,000*	2,686
	PCATSP2	39.70	5.17	27.8	55	3,663.7	3,179
	PCATSP2R	39.70	1.56	27.8	55	3,236.7	5,496
	PCATSP2R⁻	39.00	1.31	29.1	55	10,000*	34,285
	PCATSP3	40.15	8.06	27.0	55	2,785.9	1,617
	PCATSP4	31.39	0.36	42.9	55	72.4	1,373
	PCATSP5	**44.45**	12.77	19.2	55	791.0	268
	PCATSP6	**44.45**	13.67	19.2	55	791.0	268
esc25	PCATSP-SSB1	1610.96	3.41	4.2	1681	81.9	37
(z_{IP} =1681)	PCATSP-SD	1552.89	0.27	7.6	1681	95.5	3,379
	PCATSP-GP	1537.94	0.34	8.5	1681	211.0	650
	PCATSP-GP1	1540.17	0.41	8.4	1681	181.0	344
	PCATSP-GP2	1543.33	0.28	8.2	1681	133.0	251
	PCATSP-GP3	1548.29	0.48	7.9	1681	1,295.9	559
	PCATSP0	1607.72	3.13	4.4	1681	113.6	25
	PCATSP1	1649.11	376.32	1.9	1681	1,896.7	18
	PCATSP2	1649.11	124.93	1.9	1681	830.8	6
	PCATSP2R	1637.29	12.23	2.6	1681	84.7	13
	PCATSP2R⁻	1549.12	9.92	7.8	1681	1,081.4	394
	PCATSP3	1661.44	259.79	1.2	1681	2,650.0	8
	PCATSP4	1644.04	6.27	2.2	1681	56.5	8
	PCATSP5	**1681.00**	109.83	0	1681	83.6	0
	PCATSP6	**1681.00**	143.52	0	1681	105.0	0
esc47	PCATSP-SSB1	1242.37	378.05	3.5	2368	10,000*	0
(z_{IP} =1288)	PCATSP-SD	1209.16	1.69	6.1	1288	2,592.6	9,519
	PCATSP-GP	1214.83	14.30	5.7	1288	10,000*	397
	PCATSP-GP1	1214.83	95.05	5.7	2023	10,000*	150
	PCATSP-GP2	1215.94	20.59	5.6	3047	10,000*	250
	PCATSP-GP3	1216.00	159.17	5.6	-**	10,000*	22
	PCATSP0	1219.00	389.47	5.4	1305	10,000*	43
	PCATSP1	1252.45	633,913.00	2.8	-**	10,000*	0
	PCATSP2	1252.45	466,216.00	2.8	-**	10,000*	0
	PCATSP2R	1252.29	802.21	2.8	1305	10,000*	29
	PCATSP2R⁻	1232.00	516.41	4.3	-**	10,000*	0
	PCATSP3	1253.54	137,350.00	2.7	-**	10,000*	0
	PCATSP4	1242.75	489.57	3.5	2203	10,000*	0
	PCATSP5	**1263.02**	104,406.00	1.9	-**	10,000*	0
	PCATSP6	**1263.02**	111,281.00	1.9	-**	10,000*	0

- * Exceeded time limit (limited to 10,000 cpu seconds).
- ** No integer solution was obtained within the time limit.

Appendix III. Computational Comparison of Various PCATSP Formulations Using Randomly Generated Test Problems

Table A-3 displays the results obtained pertaining to the lower bound value (z_{LP}) derived by solving the LP relaxation, the cpu time (seconds) required to compute z_{LP}, the percentage gap between z_{LP} and optimal integer solution z_{IP}, the cpu time (seconds) required to obtain the best integer solution, and the number of branching nodes.

Table A- 3 Computational Results for the Various PCATSP Formulations Using Randomly Generated Test Problems

Problem	Formulation	z_{LP}	CPU time (LP)	Gap (%)	Best Integer	CPU time (IP)	Nodes
19_1 (z_{IP} =196)	PCATSP-SSB1	185.00	0.23	5.6	196	6.8	17
	PCATSP-SD	172.39	0.06	12.0	196	3.75	278
	PCATSP-GP	167.75	0.10	14.4	196	83.8	1,744
	PCATSP-GP1	168.33	0.09	14.1	196	67.1	788
	PCATSP-GP2	167.75	0.09	14.4	196	32.1	376
	PCATSP-GP3	170.00	0.22	13.3	196	66.4	223
	PCATSP0	174.67	0.13	10.9	196	11.2	17
	PCATSP1	182.30	24.94	7.0	196	259.4	20
	PCATSP2	182.30	20.17	7.0	196	137.7	17
	PCATSP2R	181.00	1.03	7.7	196	11.2	10
	PCATSP2R$^-$	174.20	0.75	11.1	196	17.3	86
	PCATSP3	183.62	17.55	6.3	196	160.1	7
	PCATSP4	187.55	0.33	4.3	196	6.8	11
	PCATSP5	**194.73**	19.42	0.6	196	34.6	0
	PCATSP6	**194.73**	24.16	0.6	196	33.0	0
19_2 (z_{IP} =272)	PCATSP-SSB1	230.83	0.16	15.1	272	24.0	185
	PCATSP-SD	219.06	0.06	19.5	272	11.3	958
	PCATSP-GP	192.20	0.10	29.3	272	404.1	7,264
	PCATSP-GP1	194.40	0.08	28.5	272	480.4	7,893
	PCATSP-GP2	194.67	0.09	28.4	272	475.0	8,452
	PCATSP-GP3	195.43	0.16	28.2	272	1,896.1	10,900
	PCATSP0	232.33	0.06	14.6	272	7.9	14
	PCATSP1	253.35	14.60	6.9	272	227.6	31
	PCATSP2	253.35	13.19	6.9	272	121.2	39
	PCATSP2R	250.52	1.74	7.9	272	23.4	47
	PCATSP2R$^-$	201.85	1.13	25.8	272	1,102.0	4,760
	PCATSP3	258.96	12.64	4.8	272	109.1	9
	PCATSP4	258.50	0.14	5.0	272	6.6	18
	PCATSP5	**268.45**	11.97	1.3	272	73.8	24
	PCATSP6	**268.45**	10.00	1.3	272	82.4	2
19_3 (z_{IP} =302)	PCATSP-SSB1	298.60	0.14	1.1	302	0.3	0
	PCATSP-SD	257.08	0.05	14.9	302	12.7	1,332
	PCATSP-GP	260.86	0.02	13.6	302	15.4	555
	PCATSP-GP1	263.50	0.05	12.7	302	13.7	276
	PCATSP-GP2	265.17	0.06	12.2	302	27.6	761
	PCATSP-GP3	267.50	0.13	11.4	302	41.3	370
	PCATSP0	274.36	0.05	9.2	302	2.5	4
	PCATSP1	298.90	6.63	1.0	302	9.3	0
	PCATSP2	298.90	3.63	1.0	302	9.3	0
	PCATSP2R	298.72	0.59	1.1	302	2.3	0
	PCATSP2R$^-$	271.50	0.72	10.1	302	22.5	51
	PCATSP3	300.99	5.06	0.3	302	12.4	0
	PCATSP4	299.22	0.14	0.9	302	0.2	0
	PCATSP5	**302.00**	1.47	0.0	302	1.2	0
	PCATSP6	**302.00**	1.14	0.0	302	1.2	0

Table A-3 Computational Results for the Various PCATSP Formulations Using Randomly Generated Test Problems (con't)

Problem	Formulation	z_{LP}	CPU time (LP)	Gap (%)	Best Integer	CPU time (IP)	Nodes
24_1 (z_{IP}=186)	PCATSP-SSB1	178.60	1.16	4.0	186	37.2	21
	PCATSP-SD	157.13	0.08	15.5	186	45.7	2,044
	PCATSP-GP	155.83	0.24	16.2	186	636.0	4,653
	PCATSP-GP1	155.83	0.22	16.2	186	784.9	3,982
	PCATSP-GP2	156.75	0.23	15.7	186	795.3	5,016
	PCATSP-GP3	156.75	0.34	15.7	186	6,794.6	8,925
	PCATSP0	176.50	0.36	5.1	186	151.3	145
	PCATSP1	177.73	144.19	4.4	186	2,488.0	8
	PCATSP2	177.73	64.38	4.4	186	880.0	10
	PCATSP2R	177.25	4.41	4.7	186	113.6	19
	PCATSP2R$^-$	159.33	3.63	14.3	186	2,452.9	1,850
	PCATSP3	178.92	122.17	3.8	186	1,103.0	9
	PCATSP4	178.60	1.19	4.0	186	28.9	8
	PCATSP5	**183.33**	115.80	1.4	186	563.1	6
	PCATSP6	**183.33**	110.61	1.4	186	614.4	2
24_2 (z_{IP}=206)	PCATSP-SSB1	183.83	0.30	10.8	206	25.3	34
	PCATSP-SD	157.78	0.17	23.4	206	42.2	1,823
	PCATSP-GP	150.50	0.14	26.9	206	1,158.5	7,243
	PCATSP-GP1	150.50	0.16	26.9	206	1,090.6	6,226
	PCATSP-GP2	151.00	0.22	26.7	206	1,149.2	7,008
	PCATSP-GP3	151.00	0.30	26.7	206	3,371.1	3,475
	PCATSP0	183.00	0.19	11.1	206	27.0	13
	PCATSP1	195.76	82.00	5.0	206	1,236.8	4
	PCATSP2	195.76	94.87	5.0	206	651.8	20
	PCATSP2R	186.25	5.16	9.6	206	108.3	15
	PCATSP2R$^-$	152.53	5.36	26.0	206	1,066.1	655
	PCATSP3	200.72	83.67	2.6	206	517.1	6
	PCATSP4	193.43	0.38	6.1	206	13.4	13
	PCATSP5	**205.39**	54.23	0.3	206	129.2	0
	PCATSP6	**205.39**	52.16	0.3	206	170.7	0
24_3 (z_{IP}=441)	PCATSP-SSB1	412.88	0.17	6.4	441	13.3	35
	PCATSP-SD	314.42	0.16	28.7	441	149.9	6,886
	PCATSP-GP	345.00	0.11	21.8	441	809.4	6,555
	PCATSP-GP1	345.00	0.14	21.8	441	1,864.0	10,405
	PCATSP-GP2	348.86	0.17	20.9	441	1,218.7	7,558
	PCATSP-GP3	348.86	0.28	20.9	441	6,327.3	14,447
	PCATSP0	386.27	0.13	12.4	441	16.5	16
	PCATSP1	425.09	41.18	3.6	441	356.2	60
	PCATSP2	425.09	39.78	3.6	441	248.3	54
	PCATSP2R	415.50	3.81	5.8	441	75.3	35
	PCATSP2R$^-$	351.00	3.08	20.4	441	6,463.4	5,072
	PCATSP3	429.26	31.47	2.7	441	175.9	81
	PCATSP4	413.45	0.19	6.2	441	11.3	22
	PCATSP5	**441.00**	17.56	0.0	441	17.9	0
	PCATSP6	**441.00**	15.38	0.0	441	16.2	0

References

[1] Ali, A.I. and J.L. Kennington, The asymmetric m-traveling salesmen problem: a duality based branch-and-bound algorithm, *Discrete Applied Mathematics*, 13 (1986), 259-276.

[2] Appelqvist, P. and J.-M. Lehtonen, Combining optimisation and simulation for steel production scheduling, *Journal of Manufacturing Technology Management*, 16 (2) (2005), 197-210.

[3] Applegate, D. and W. Cook, A computational study of the job-shop scheduling problem, *ORSA Journal on Computing*, 3(2) (1991), 149-156.

[4] Assaf, I., M. Chen, and J. Katzberg, Steel production schedule generation, *International Journal of Production Research*, 35(2) (1997), 467-477.

[5] Baker, K.R., *Introduction to Sequencing and Scheduling*, John Wiley, New York, N.Y. 1974.

[6] Balas, E., Machine sequencing via disjunctive graphs: an implicit enumeration algorithm, *Operations Research*, 17(6) (1969), 941-957.

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

[8] Balas, E., The prize collecting traveling salesman problem, *Networks*, 19 (1989), 621-636.

[9] Balas, E., New classes of efficiently solvable generalized traveling salesman problems, *Annals of Operations Research*, 86 (1999), 529-558.

[10] Balas, E., M. Fischetti, and W. R. Pulleyblank, The precedence-constrained asymmetric traveling salesman polytope, *Mathematical Programming*, 68 (1995), 241-265.

[11] Bellmore, M. and S. Hong, Transformation of multisalesmen problem to the standard traveling salesman problem, *Journal of Association for Computing Machinery*, 21(3) (1974), 500-504.

[12] Bektas, T., The multiple traveling salesman problem: an overview of formulations and solution provedures, *Omega*, 34 (3) (2006), 209-219.

[13] Benders, J.F., Partitioning procedures for solving mixed variables programming problems, *Numerishe Mathematik*, 4 (1962), 238-252.

[14] Box, R. E. and D. G. Jr. Herbe., A scheduling model for LTV Steel's Cleveland Works' twin strand continuous slab caster, *Interfaces*, 18(1) (1988), 42-56.

[15] Carr, R.D. and G Lancia, Compact vs. exponential-size LP relaxations, *Operations Research Letters*, 30 (2002), 57-65.

[16] Chen, T.-R., T.-S. Chang, C.-W. Chen, and J. Kao, Scheduling for IC sort and test facilities with preemptiveness via Lagrangian relaxation, *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 25 (8) (1995), 1249-1256.

[17] Chen, T.-R. and T.C. Hsia, Scheduling for IC sort and test facilities with precedence constraints via Lagrangian relaxation, *Journal of Manufacturing Systems*, Vol. 16(2) (1997), 117-128.

[18] Christofides, N., A. Mingozzi, and P. Toth, Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations, *Mathematical Programming*, 20 (1981), 255-282.

[19] Dantzig, G.B., D.R. Fulkerson, and S.M. Johnson, Solutions of a large scale traveling salesman problem, *Operations Research*, 2 (1954), 393-410.

[20] Desrochers, M. and G. Laporte, Improvements and extensions to the Miller-Tucker-Zemlin subtour elimination constraints, *Operations Research Letters*, 10 (1991), 27-36.

[21] Fisher, M.L., Optimal solution of scheduling problems using Lagrange multipliers: Part I, *Operations Research*, 21 (1973), 1114-1127.

[22] Fisher, M.L., B.J. Lageweg, J.K. Lenstra, and A.H.G. Rinnooy Kan, Surrogate duality relaxation for job shop scheduling, *Discrete Applied Mathematics*, 5 (1983), 65-75.

[23] Gavish, B., A note on the formulation of the m-salesman traveling salesman problem, *Management Science*, 22(6) (1976), 704-705.

[24] Gavish, B. and K. Srikanth, An optimal solution method for large-scale multiple traveling salesmen problems, *Operations Research*, 34(5) (1986), 698-717.

[25] Gorenstein, S., Printing press scheduling for multi-edition periodicals, *Management Science*, 16(6) (1970), B373-383.

[26] Gouveia, L. and J. Pires, The asymmetric traveling salesman problem and a reformulation of the Miller-Tucker-Zemlin constraints, *European Journal of Operational Research*, 112 (1999), 134-146.

[27] Gouveia, L. and J. Pires, The asymmetric traveling salesman problem: on generalizations of disaggregated Miller-Tucker-Zemlin constraints, *Discrete Applied Mathematics*, 112 (2001), 129-145.

[28] Grotschel, M. and M. Padberg, Polyhedral computations. In: E.L. Lawler, J.K. Lenstra, A. Rinnooy Kan and D.B. Shmoys, Editors, *The Travelling Salesman Problem: A Guided Tour of Combinatorial Optimization*, Wiley, New York, NY, 1985.

[29] Grotschel, M. and M. Padberg, Polyhedral theory. In: E.L. Lawler, J.K. Lenstra, A. Rinnooy Kan and D.B. Shmoys, Editors, *The Travelling Salesman Problem: A Guided Tour of Combinatorial Optimization*, Wiley, New York, NY, 1985.

[30] Held, M. and R.M. Karp, The travelling salesman problem and minimum spanning trees, *Operations Research*, 18 (1970), 1138-1162.

[31] Held, M. and R.M. Karp, The travelling salesman problem and minimum spanning trees: Part II, *Mathematical Programming*, 1 (1971), 6-25.

[32] Hestenes, M. R. and E. Stiefel, Methods of Conjugate Gradients for Solving Linear Systems, *Journal of Research of the National Bureau of Standards, Section B* 48 (1952), 409-436.

[33] Hong, S. and M.W. Padberg, A note on the symmetric multiple traveling salesman problem with fixed charges, *Operations Research*, 25(5) (1977), 871-874.

[34] Huang, H., M.H. Wang, and M.R. Johnson, Disassembly sequence generation using neural network approach, *Journal of Manufacturing Systems*, 19(2) (2000), 73-82.

[35] Ian Christmas, The Challenges Ahead for Steel, Secretary General, IISI, An address to the *IMI Annual General Meeting*, St. Andrews Scotland, 2 June 2005.

[36] International Iron and Steel Institute (IISI), World Steel in Figures Online - Steel statistical data showing trends in the world steel industry: World Steel in Figures 2005, available at: http://www.worldsteel.org/wsif.php.

[37] Iron & Steel Industry production reports from the ISSB, ISSB Monthly World I&S Review, *World Steel Review*, February 2006, available at: http://www.steelonthenet.com/production.html.

[38] Jacobs, T. L., J. R. Wright, and A. E. Cobbs, Optimal inter-process steel production scheduling, *Computers & Operations Research*, 15(6) (1988), 497-507.

[39] Jonker, R. and T. Volgenant, An improved transformation of the symmetric multiple traveling salesman problem, *Operations Research*, 36(1) (1988), 163-167.

[40] Kosiba, E. D., J. R. Wright, and A. E. Cobbs, Discrete event sequencing as a traveling salesman problem, *Computers in Industry*, 19 (1992), 317-327.

[41] Laporte, G. and Y. Nobert, A cutting planes algorithm for the m-salesmen problem, *Journal of the Operational Research Society*, 31 (1980), 1017-1023.

[42] Lee, H. S. and S. S. Murthy, Primary production scheduling at steel-making industries, *IBM Journal of Research & Development*, 40(2) (1996), 231-253.

[43] Lim, C., *Nondifferentiable Optimization of Lagrangian Dual Formulations for Linear Programs with Recovery of Primal Solutions*, PhD Dissertation, Dept. of Industrial and Systems Engineering, VPI & SU, Blacksburg, VA, 2004.

[44] Lim, C., *C++ NDO Codes Manual*, Grado Department of Industrial and Systems Engineering, Virginia Polytechnic Institute and State University, Blacksburg, Virginia, 2004.

[45] Lim, C. and H.D. Sherali, Convergence and computational analysis for some variable target value and subgradient deflection methods, *Computational Optimization and Applications*, (in print), 2006.

[46] Lim, C. and H.D. Sherali, A trust region target value method for optimizing nondifferentiable Lagrangian duals of linear programs, *Mathematical Methods of Operations Research*, (accepted), 2006.

[47] Lopez, L., M. W. Carter, and M. Gendreau, The hot strip mill production scheduling problem: a tabu search approach, *European Journal of Operational Research*, 106 (1998), 317-335.

[48] Manne, A.S., On the job-shop scheduling problem, *Operations Research*, 8 (1960), 219-223.

[49] Mascis, A. and D. Pacciarelli, Job-shop scheduling with blocking and no-wait constraints, *European Journal of Operational Research*, 143 (2002), 498-517.

[50] Meacham, A., R. Uzsoy, and U. Venkatadri, Optimal disassembly configurations for single and multiple products, *Journal of Manufacturing Systems*, 18 (5) (1999), 311-322.

[51] Miller, C., A. Tucker, and R. Zemlin, Integer programming formulations and travelling salesman problems, *Journal of the Association for Computing Machinery*, 7 (1960), 326-329.

[52] Miller, D. L. and J. F. Pekny, Exact solution of large asymmetric traveling salesman problems, *Science*, New Series, 251 (4995) (1991), 754-761.

[53] Muth, J.F. and G.L. Thompson (eds.), *Industrial Scheduling*, Prentice-Hall, Englewood Cliffs, NJ., 1963.

[54] Nemhauser, G.L. and L.A. Wolsey, *Integer and Combinatorial Optimization*, second edition, Wiley, New York, NY, 1999.

[55] Pacciarelli, D., Alternative graph formulation for solving complex factory-scheduling problems, *International Journal of Production Research*, 40(15) (2002), 3641-3653.

[56] Panwalkar, S.S., R.A. Dudek, and M.L Smith, Sequencing research and the industrial scheduling problem, *Symposium on the Theory of Scheduling and its Applications*, S.E. Elmaghraby (editor), (1973), 29-38.

[57] Pearn, W.L., S.H. Chung, and M.H. Yang, A case study on the wafer probing scheduling problem, *Production Planning and Control*, 13(1) (2002), 66-75.

[58] Petersen, C. M., K. L. Sørensen, and R. V. V. Vidal, Inter-process synchronization in steel production, *International Journal of Production Research*, 30 (6) (1992), 1415-1425.

[59] Pinedo, M., *Scheduling: Theory, Algorithms, and Systems*, second edition, Prentice Hall, Englewood Cliffs, N.J. 2002.

[60] Rao, M.R., A note on the multiple traveling salesman problem, *Operations Research*, 28(3) (1980), 628-632.

[61] Sarin, S.C., H.D. Sherali, and A. Bhootra, New tighter polynomial length formulations for the asymmetric traveling salesman problem with and without precedence constraints, *Operations Research Letters*, 33(1) (2005), 62-70.

[62] Sarin, SC, H.D. Sherali, and A. Bhootra, A precedence constrained asymmetric traveling salesman model for disassembly optimization, *IIE Transactions*, 38 (2006), 223-237.

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

[64] Sherali, H.D. and W.P. Adams, A hierarchy of relaxations and convex hull characteristics for mixed-integer zero-one programming problems, *Discrete Applied Mathematics*, 52 (1994), 83-106.

[65] Sherali, H.D. and W.P. Adams, *A Reformulation-Linearization Technique for Solving Discrete and Continuous Nonconvex Problems*, Kluwer Academic, Boston, Mass, 1999.

[66] Sherali, H. D., G. Choi, and C. H. Tuncbilek, A Variable Target Value Method for Nondifferentiable Optimization, *Operations Research Letters*, 26(1) (2000), 1-8.

[67] Sherali, H.D. and P.J. Driscoll, On tightening the relaxations of Miller-Tucker-Zemlin formulations for asymmetric traveling salesman problems, *Operations Research*, 50(5) (2002), 656-669.

[68] Sherali, H.D. and C. Lim, On embedding the volume algorithm in a variable target value method, *Operations Research Letters*, 32 (5) (2004), 455-462.

[69] Sherali, H. D. and O. Ulular, A Primal-Dual Conjugate Subgradient Algorithm for Specially Structured Linear and Convex Programming Problems, *Applied Mathematics and Optimization*, 20(2) (1989), 193-221.

[70] Svestka, J.A. and V.E. Huckfeldt, Computational experience with an m-salesman traveling salesman algorithm, *Management Science*, 19(7) (1973), 790-799.

[71] Tamura, R., M. Nagai, Y. Nakagawa, T. Tanizaki, and H. Nakajima, Synchronized scheduling method in manufacturing steel sheets, *International Transaction on Operational Research*, 5(3) (1998), 189-199.

[72] Tang, L, J. Liu, A. Rong, and Z. Yang, A multiple traveling salesman problem model for hot rolling scheduling in Shanghai Baoshan Iron & Steel Complex, *European Journal of Operational Research*, 124 (2000), 267-282.

[73] Uzsoy, R., L.A. Martin-Vega, C.-Y. Lee, and P.A. Leonard, Production scheduling algorithms for a semiconductor test facility, *IEEE Transactions on Semiconductor Manufacturing*, 4(4) (1991), 270-280.

[74] Wilbrecht, J.K. and W.B. Prescott, The influences of setup time on job shop performance, *Management Science*, 16 (1969), B274-B280.

[75] Wong, R.T., Integer programming formulations of the traveling salesman problem, *Proceedings of the IEEE International Conference on Circuits and Computers*, Part I, New York, NY, 149-152, 1980.

Vita

Pei-Fang (Jennifer) Tsai was born as the first child of Sho-Chin Tsai-Fang and Li-Chao Tsai in Kaohsiung City, Taiwan. She received her B.S. and M.S degrees in Industrial Engineering (IE) at Tunghai University, Taiwan in June 1995 and 1997, respectively. While completing her B.S. degree, Pei-Fang stretched her musical interests by joining the Philharmonic Choir in Department of Music at Tunghai University and serving in the Christmas Mess as a choir member. She graduated with a rank of 5th in her class.

After graduation, she worked as an associate researcher in the Industrial Technology Research Institute (ITRI) located in Sinchu, Taiwan, which is a national-level, non-profit R&D organization engaging in applied research and technical service. She participated in a 10-year, government-funded project aimed to the development of shop-floor oriented information systems, referred to as manufacturing execution systems (MES), for small and medium sized companies. She was in charge with the development as well as the implementation of the stochastic quality control (SQC) system as a consultant with IT developers in semiconductor packaging and assembly companies.

Pei-Fang received her Ph.D. degree in the Grado Department of Industrial and Systems Engineering (ISE) at Virginia Polytechnic Institute and State University in May 2006. During her Ph.D. studies, she prepared herself for her academia career by working as a teaching assistant and research assistant in the ISE department. Her research interests are in applied mathematical programming and optimization, especially in the areas of manufacturing production planning and scheduling and transportation logistics. She will join the Electronics Manufacturing Research and Services (EMRS) organization, a prominent research arm of the Department of Systems Science and Industrial Engineering (SSIE) in the Thomas J. Watson School of Engineering and Applied Sciences at the State University of New York at Binghamton (SUNY) as a Research Assistant Professor.